AFRL-RI-RS-TR-2018-041

FORMAL MODELING, MONITORING, AND CONTROL OF EMERGENCE IN DISTRIBUTED CYBER-PHYSICAL SYSTEMS

UNIVERSITY OF TEXAS AT ARLINGTON

FEBRUARY 2018

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

NOTICE AND SIGNATURE PAGE

AFRL-RI-RS-TR-2018-041 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

/ S /	**/ S /**
WILLIAM D. LEWIS	JULIE BRICHACEK
Work Unit Manager	Chief, Information Systems Division
	Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*
FEB 2018

2. REPORT TYPE
FINAL TECHNICAL REPORT

3. DATES COVERED *(From - To)*
APR 2015 – APR 2017

4. TITLE AND SUBTITLE

FORMAL MODELING, MONITORING, AND CONTROL OF EMERGENCE IN DISTRIBUTED CYBER-PHYSICAL SYSTEMS

5a. CONTRACT NUMBER
FA8750-15-1-0105

5b. GRANT NUMBER
N/A

5c. PROGRAM ELEMENT NUMBER
62788F

6. AUTHOR(S)

Taylor T. Johnson

5d. PROJECT NUMBER
S2MA

5e. TASK NUMBER
VU

5f. WORK UNIT NUMBER
TA

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Vanderbilt University
1025 16th Ave. S.
Nashville, TN 37212

University of Texas at Arlington
500 UTA Blvd
Arlington, TX 76019-0015

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Research Laboratory/RISC
525 Brooks Road
Rome NY 13441-4505

10. SPONSOR/MONITOR'S ACRONYM(S)
AFRL/RI

11. SPONSOR/MONITOR'S REPORT NUMBER
AFRL-RI-RS-TR-2018-041

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This project studied emergent behavior in distributed cyber-physical systems (DCPS). Emergent behavior does not a priori appear in the descriptions of such systems. The approach undertaken is to characterize this perspective as invalid: any behavior not a prior specified by the requirements and specifications of a system is emergent. With this perspective and using formal methods, formal verification with reachability analysis and inductive invariance, as well as architectural runtime monitoring and runtime assurance, this project developed and demonstrated novel ways to specify, verify, monitor, and control behavior in DCPS, such as groups of unmanned autonomous systems (UASs).

15. SUBJECT TERMS
Formal methods; emergent behavior; learning; autonomy; cyber-physical systems; distributed systems

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
					WILLIAM D. LEWIS
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER (*Include area code*)
U	U	U	UU	367	

Table of Contents

List of Figures

List of Tables

1. Summary

This final technical report covers the two-year duration of this project, from April 16, 2015 through April 15, 2017.

As Air Force warfighting missions incorporate increased distributed autonomy, emergent global behavior may arise from interactions between individual autonomous agents. Example Air Force systems that may in the coming years incorporate increased autonomy resulting in little-to-no direct human monitoring and intervention include drone (UAV) swarms [1] and satellite constellations [2]. Novel methods are needed to ensure such distributed cyber-physical systems (DCPS) have trusted assurance to meet their mission requirements and only their mission requirements in spite of potential emergent distributed behavior, attacks, and failures. Understanding distributed emergence and being able to respond to it through trusted and assured responses will allow warfighters to continue fighting and adapting through engagements, enabling strategic agility in Air Force missions. Ultimately, developing theoretical and practical tools for understanding and responding to the fundamental phenomena of emergence will enable the Air Force goal to fly, fight, and win … in air, space, and cyberspace.

This project suggested and developed the use of scalable formal methods in mission (1) specification and verification, (2) runtime monitoring, and (3) trusted and assured control, all conducted in conjunction with (4) a rigorous evaluation on DCPS with prototypical features of modern Air Force systems such as UAV swarms and satellite constellations. The primary research objectives undertaken were to:

- **Objective 1:** Develop scalable automated formal verification methods for specifying and verifying trusted global DCPS mission behaviors along with distributed emergent behavior, alleviating state-space explosion by exploiting symmetries.

- **Objective 2:** Develop scalable runtime verification and monitoring methods relying on both formal tools and heuristic systems to detect emergent behaviors and violations of global mission specifications during mission operation.

- **Objective 3:** Develop runtime assurance (RTA)-like trusted control methods for these distributed systems building upon the foundational theory of self-stabilization of distributed systems [3-6] and the Simplex architecture [7] to ensure mission specifications are maintained in spite of emergent distributed behaviors at execution time.

- **Objective 4:** Evaluate the formal specification and verification, runtime monitoring, and control methods developed in the other objectives on challenging DCPS case studies with Air Force relevance, particularly swarm robot systems.

2. Introduction

2.1. Motivation

Physical systems are becoming increasingly dependent upon computers and software, such as in emerging embedded and cyber-physical systems (CPS), where networked software interacts with physical processes. For instance, typical modern cars utilize dozens-to-hundreds of microprocessors, many communications buses, and a complex interconnection between sensors, actuators, and processors [8-10]. In the design and development process for most engineered systems today (including CPS), the vast majority of resources are devoted to ensuring systems meet their specifications [11, 12]. In spite of significant technical advances for design verification and validation—such as model checking, hardware-in-the-loop testing, automatic test case generation for software, and sophisticated simulators—there are frequent safety recalls across CPS industries due to problems between cyber and physical subcomponents. For example, the Consumer Product Safety Commission (CPSC) has recalled between 2010-2012 fire alarm and control systems from Bosch, Tyco-Grinnel, and Honeywell for failure to sound alarms and/or notify fire departments [13-15], the Food and Drug Administration (FDA) has reported the leading cause of recent medical device recalls are cyber-related (tied with manufacturing defects) [16, 17], and the National Highway Traffic Safety Administration (NHTSA) has recalled hundreds of thousands of 2004-2005 and 2010-2014 Toyota Priuses due to drivetrain software problems causing unexpected stalls [18, 19] and millions of 2005-2010 Hondas due to electronic control model software causing transmission damage [20]. Given that such recalls are due to increased risk of physical safety (and not yet, e.g., for privacy issues), all such problems are inherently cyber-physical. As future networked systems like robot swarms, the smart grid, satellite constellations, and the intelligent transportation system increasingly couple distributed agents together, emergent behavior will be seen to spontaneously arise. Demonstrated areas of distributed emergence through local interaction in natural and engineered systems include fish schools [21], herds [22], highways [23, 24], swarm robotics [25-27], and distributed computing [4, 28, 29].

2.2. Air Force and Department of Defense Relevance

Air Force Relevance for Strategic Agility: In the July 2014 report, "America's Air Force: A Call to the Future," Secretary of the Air Force Deborah Lee James outlines a three decades long strategic plan for the Air Force, centered around the theme of strategic agility. In this strategy, two technical areas of relevance to this project are highlighted, namely autonomous systems and unmanned systems. From a technical standpoint for unmanned systems, strategic agility will enable systems with little human supervision to "swarm, suppress, deceive, or destroy." For autonomous systems, strategic agility will enable moving from today's systems that are "able to execute a set of pre-programmed functions" to tomorrow's systems that "will be better able to react to their environment and perform more situational-dependent tasks as well as synchronized and integrated with other autonomous systems." The work completed through this project brings a formal perspective to what it means for systems to have emergent behavior, such as what may arise in the challenging environments of the battlefield.

Department of Defense Relevance: In the January 2013 report "Resilient Military Systems and the Advanced Cyber Threat" from the Defense Science Board, a number of broad cyber challenges and opportunities are outlined in current and future defense systems. Specific recommendations of the task force report include "use of emerging technology developments for system resilience, such as trust anchors, minimal functionality components, simplified operating systems, developing a means to verify compromise of fielded systems contributing to critical missions, creating trust in systems built with un-trusted components, and restoring to a known state."

Many Air Force and DoD projects are currently underway related to these areas. For example, the High-Assurance Cyber Military Systems (HACMS) projects aims in part to develop verified components, the BEDROCK project, high-assurance microprocessors are in development, techniques for determining computer component intrusion and counterfeiting, etc. Verified operating systems like seL4 and verified optimizing compilers like CompCert are new seminal results toward this goal. However, the work completed in this project is uniquely differentiated from all existing work in several ways. First, this work focuses on emergent behaviors and properties that may arise in distributed systems, while most if not all of these other projects and existing approaches focus on non-distributed systems. We are not yet to the state where distributed systems may be fully verified (e.g., at every layer of the OSI network model), although progress is being made as outlined above (and in e.g., verification of cryptographic protocols, key exchanges, etc.). Second, the work of this project is not a clean-slate approach like HACMS and BEDROCK. The results of this project may operate within the constraints defined by existing development environments practices, and the reality that for a variety of reasons (e.g., budgetary), there is additional use of commercial off-the shelf components (COTS) in military systems. To operate within these constraints, we investigated both fully formal approaches and semi-formal approaches augmented with heuristic approaches.

2.3. High-Level Technical Summary

The underlying formal, mathematical framework used in this project is that of ***hybrid automata*** [30], which are finite-state machines augmented with real-valued variables that evolve continuously over intervals of real time. Asynchronous networks composed of hybrid automata [31] are useful for modeling distributed systems that interact with the physical world, such as robot swarms [5, 6], air traffic control systems [32, 33], autonomous satellites and constellations [2], and distributed electrical microgrids [34]. Desired emergent behaviors include phenomena like flocking, while undesired emergent behavior may lead to catastrophic mission failure.

Objective 1: Scaling Formal Specification and Verification for Emergence in DCPS

Objective 1 developed design-time formal specification and verification methods for emergence in DCPS modeled as networks of hybrid automata with linear and nonlinear dynamics. For specifying emergence, new specification languages for CPS using hyperproperties were developed allowing specification of frequency-domain behavior and real-time, real-valued behaviors through hyperproperties for signal temporal logic (HyperSTL) [35-37]. The Passel verification tool [31, 38], in conjunction with a small model theorem [39], an invariant synthesis procedure [32], and a symmetry-reduction reachability method [40], enabled the first fully automatic verification of safety (aircraft separation) for the Small Aircraft Transportation System (SATS) landing protocol (a part of the NASA/FAA NextGen program [41-48]). Through this objective, we built upon these approaches for addressing the state-space explosion problem to scale verification methods to larger DCPS than previously possible, as well as developed a new verification tool, HyST [49].

Objective 2: Detecting Emergence at Runtime: Specification-Based Runtime Monitoring

Objective 2 developed formal and heuristic runtime monitoring verification methods for emergence in DCPS, using both model-based and model-free approaches. Model-based methods rely on formal methods tools and inherently are subject to scalability problems, while model-free approaches are heuristic, as they are both unsound and incomplete, but scale better. Together, the methods rely on monitoring asynchronous distributed and hybrid systems at runtime and in real-time, and build upon both model-free and model-based approaches developed by our group [1, 7, 50]. For DCPS, we extended an invariant inference tool called Hynger (HYbrid iNvariant GEneratoR) [50] that instruments arbitrary MathWorks Simulink/Stateflow (SLSF) models to generate candidate invariants over input and output variables [51].

Objective 3: Assured Control in Spite of Emergence with Real-Time Reachability and Self-Stabilization for Distributed Simplex

Objective 3 developed control methods to ensure desirable or avoid undesirable emergent distributed behavior at runtime, by leveraging the Simplex-based RTA framework using real-time reachability of networks of hybrid automata in conjunction with self-stabilization [3-6, 52, 53] of the distributed system. Monitoring predicates over physical variables and their continuous evolution over time was performed with real-time hybrid systems reachability and runtime monitoring of emergent behavior specified and detected in Objectives 1 and 2.

Objective 4: Evaluating Analysis, Monitoring, and Control of Emergence in DCPS

Objective 4 is evaluating the novel methods for distributed emergence developed in the previous objectives. We performed analytical analysis, simulations, laboratory experiments, and demonstrations using a swarm of autonomous agents, particularly commercially available quadrotor drones. Typical safety properties that arise are collision avoidance and convergence to some desired configuration and/or location [52], and emergent properties may be consensus, flocking, or unwanted oscillatory movements due to failures, attacks, communication delays, etc. Our results in similar studies include verification of autonomous satellite maneuvers [2], flocking in swarm robotics in spite of failures [5, 6], and planar robotics [52].

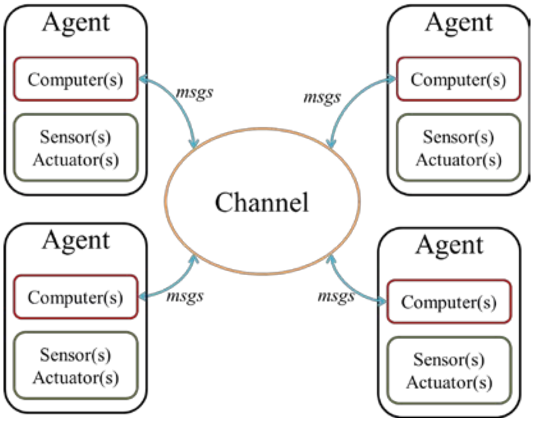


Figure 1: High-level overview of the DCPS modeling framework, where each agent (participant) in the distributed system is modeled as a hybrid automaton, and a network is composed of these automata that may communicate through a potentially lossy and adversarial channel.

3. Methods, Assumptions, and Procedures

3.1. Overview

The objectives summarized in the previous section were undertaken through the following technical procedures and methods.

3.2. Technical Procedures

3.2.1. Scaling Formal Specification and Verification for Emergence in DCPS

We first developed design-time formal specification and verification methods for emergence in DCPS modeled as networks of hybrid automata with linear and nonlinear dynamics. We developed a verification framework for modeling DCPS as networks of hybrid automata that interact through discrete transitions [31, 32, 39, 40]. The Passel verification tool [31, 38], in conjunction with a small model theorem [39] and an invariant synthesis procedure [32], enabled the first fully automatic verification of safety (aircraft separation) for the Small Aircraft Transportation System (SATS) landing protocol (a part of the NASA/FAA NextGen program [41-48]). Extending Passel and its theoretical framework to emergence properties for swarm robotics first requires developing formal definitions and specifications of emergence properties in DCPS.

3.2.1.1. Formally Defining and Specifying Emergent Behavior in DCPS

The approach is to specify emergence as sets of invariant properties over the local states of individual automata, to describe the global behavior of the entire distributed systems. For example, invariants allow specifying either creation or absence of emergence of consensus or flocking behavior. DCPS are naturally parameterized by a number of interacting agents, for instance, the number of robots in a swarm. We define absence and presence of emergence properties using invariants, integrate the formal specification of emergence into an extension of the Passel verification tool [31, 38] with a new software tool HyST [49], and evaluate synthesizing implementations in a correct-by-construction manner [54].

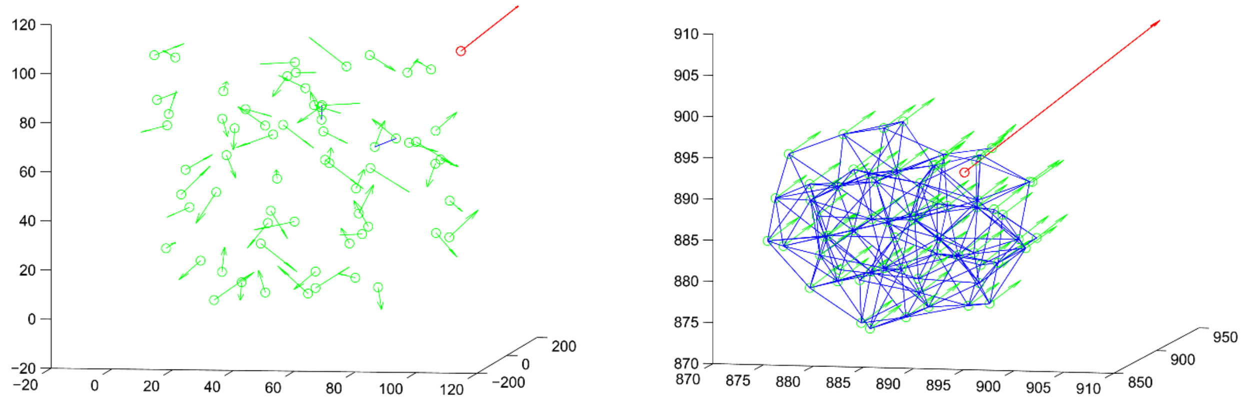


Figure 2: Example of emergent flocking behavior in three dimensions with a system of N = 64 agents. The left frame is at an initial condition and the right frame illustrates the flocking formation after 36 seconds of runtime. The agent positions are denoted by green circles, their velocities by green vectors, and a red vector indicates their desired heading. Blue lines between agents are drawn if their distances are approximately spaced by some desired flocking spacing r_f.

Figure 2 shows emergent flocking behavior in a simulation of the Olfati-Saber algorithms [25-27] that rely only on local communication between the agents. Flocking is not specified anywhere in the system description, instead, it emerges dynamically as a property of the system over its execution. One definition of flocking is that all agents are spaced equally from all their neighbors, which may be specified mathematically as:

$$\forall i \in [N], \forall j \in C_i \colon \left\| x_i - x_j \right\| = r_f, \tag{1}$$

where i, j come from a set of agent identifiers $[N] \triangleq \{1, \ldots, N\}$, x_i, x_j are real vectors of an appropriate dimensionality (e.g., 3 for the example of Figure 2), $\| \cdot \|$ is an appropriate norm (e.g., say the 2-norm), C_i is a set of communication neighbors of i (e.g., $C_i \triangleq \{j \in [N] \colon \left\| x_i - x_j \right\| \leq r_c\}$ for some communication radius r_c), and $r_f > 0$ is some desired flocking spacing [1, 5, 6]. Note that non-ideal spacing may easily be incorporated, e.g., to define a flock as states where agents are approximately spaced by r_f, such as $r_f \pm \epsilon_f$ for some small ϵ_f. Control algorithms to enable such emergent behavior do not a prior specify anything about the behavior, rather it arises spontaneously. Other emergent properties of interest for such systems include collision avoidance, which may be specified in a similar format, such as:

$$\forall i, j \in [N] \colon \left\| x_i - x_j \right\| \geq r_s, \tag{2}$$

where all quantities are as before and $r_s \leq r_f$ is a desired spacing amount. Note that these emergent behaviors are potentially in conflict with one another: flocking mandates agents come sufficiently close together, while safety mandates agents do not come too close together.

From a specification standpoint of these two different forms of emergent behavior described in (1) and (2), there are several similarities. First, the class of formulas these specifications come from is quite similar. These are both specified using universal quantification

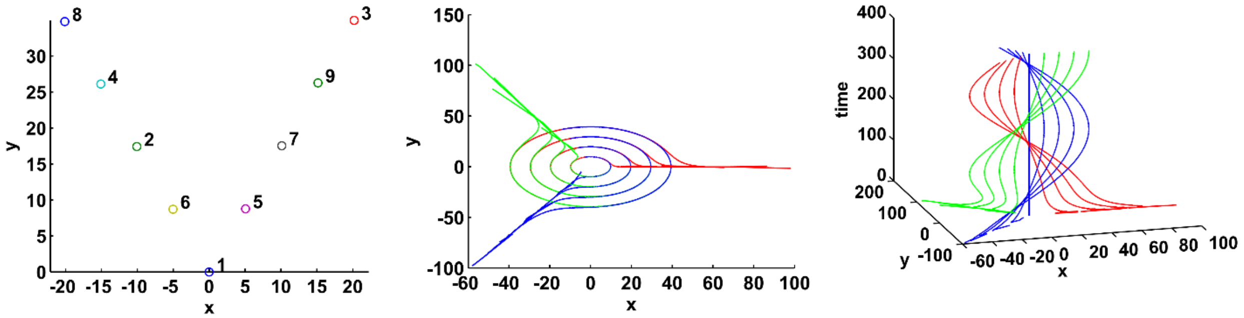


Figure 4: Alternative specifications of flocking emergence exist like bird vees. This is a planar scenario created by adjoining two one-dimensional flocks (platoons) about an appropriate angle. The middle figure shows a phase space plot of the trajectories of all agents, and the right figure shows the planar coordinates of all agents as they evolve over time while moving and rotating in the plane. By composing formally verified primitives (the exponentially stable one-dimensional flocking algorithm), sophisticated and verified planar formation control is achieved. High-level mission specifications and flock formation parameters (such as the angles, where to move, etc.) may be specified in a temporal logic like linear temporal logic (LTL).

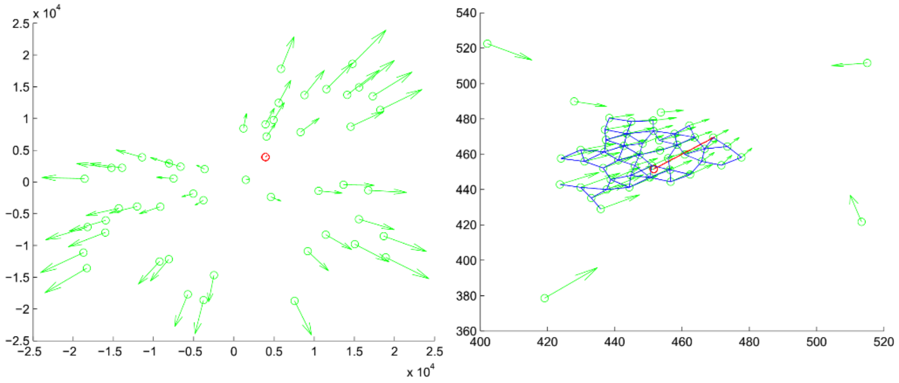


Figure 3: The left image shows divergent emergent behavior when trying to use distributed flocking control algorithms with realistic system constraints, particularly (1) actuator saturation, (2) asynchrony, and (3) communication delays, and the right image shows partial emergence of flocking for these factors.

followed by a quantifier-free formula over reals. Both of these specifications of emergent behavior almost fall into the restricted class of first-order logic (FOL) supported in the theoretical framework developed for uniform verification of safety properties in networks of hybrid automata, with automated reasoning methods implemented in the Passel software tool [31, 32, 39]. To highlight one subtlety, note that an alternative way to represent the set of communication neighbors of an agent is using a set-valued variable, i.e., an array.

However, extensions are needed to support planar and three-dimensional specifications of flocking, extensions to the restricted class of FOL supported by the small model theorem [39] exploited by Passel. Specifically, the specification of the two-norm is a polynomial expression over the reals, while Passel has only been used so far on linear expressions. Thus, a first objective is to extend Passel to support polynomial expressions. Next, realistic systems have continuous dynamics specified by linear or nonlinear ordinary differential equations (ODEs), while the modeling language supported by Passel does not currently allow this. Additionally, an extension of the small model theorem for these scenarios is required, as it also only allows linear expressions, while the solutions of ODEs may generally involve special functions and transcendentals. This extension to the theoretical basis of Passel was made and integrated within HyST

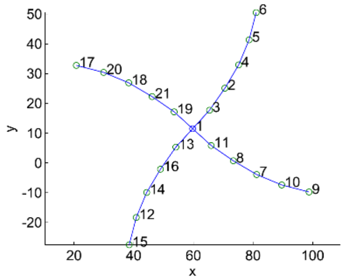


Figure 5: Emergent flocking with four groups of agents using platooning algorithms.

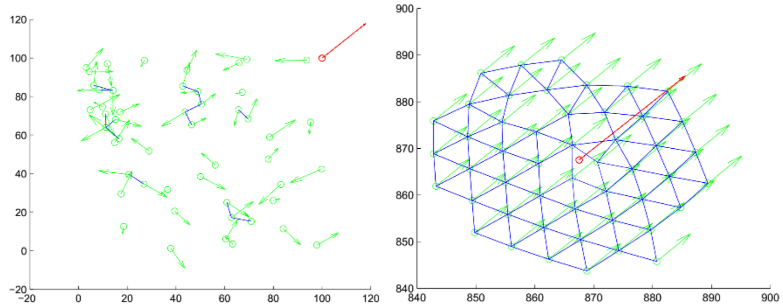


Figure 6: Emergent planar flocking behavior under ideal conditions required by existing distributed control algorithms without attacks, failures, control/actuation saturation, asynchrony, or communication failures.

[49]. This extension is feasible, under the assumption that the continuous dynamics of an agent i do not directly depend upon those of an another agent j. That is to say, their continuous interactions are decoupled. They may however interact discretely, through for instance communication or computer-sampled sensing. This together leads to the next approach, of extending the modeling language and theoretical basis of Passel to support both decoupled linear and nonlinear ODEs.

3.2.1.2. *Formal Verification using State-Space Reductions in Hybrid Automata Networks*

Previous limitations of verification methods for DCPS required each automaton in the network to have rectangular dynamics ($\dot{x} \in [a, b]$ for real constants $a \leq b$). While many systems' dynamics may be reasonably over-approximated as rectangular differential inclusions, it is critical to extend the framework and results to support linear and nonlinear differential equations. The Passel verification tool and its theoretical basis was extended within HyST to support DCPS with linear and nonlinear continuous dynamics, enabling it to realistically specify and verify swarm robotics case studies with emergence by exploiting symmetry-reduction methods for reachability [31, 40, 55-63] and small model theorems [39] for proving inductive invariants to establish. Since these methods are sound and consider all system behaviors and permutations, they have the capability to establish the presence or absence of emergence over the evolution of these DCPS.

The main technical challenge in utilizing such methods (for any formal model) is the state-space explosion problem (referred to as the "curse of dimensionality" in other fields) [31, 55-57, 64-68], which is that the size of the state-space grows exponentially in the number of components (see Figure 7). For example, small model theorems [31, 39, 67, 69-75] allow for formally verifying safety and liveness properties of arbitrarily large parameterized networks of communicating automata using finite (and typically small) equivalent systems. The "small model" here refers to the size of models in the formal logic sense that are necessary to consider in deductive proofs. That is, a model is a satisfying assignment to a sentence, and the size refers to the largest size of satisfying assignments that need be considered, and not to the size of the system model itself, although there is clearly a relationship between the two.

State-space explosion is a challenging problem in verification, and in [39] we developed a small model method for verifying safety properties of arbitrarily large networks of hybrid automata by verifying finite networks. For example, in an air traffic control system, each aircraft may be modeled as a hybrid automaton and a safety specification is that no two aircraft ever come too close to one another to establish that aircraft never collide. A major focus of this research community is to develop mathematical and software tools to verify that CPS design models meet their requirements. Of course, automation is challenging for a variety of reasons, such as the state-space explosion problem and the combinations of discrete and continuous dynamics. Significant effort was spent developing a software tool called Passel—a collective noun meaning a large group of indeterminate number—for automatic verification of parameterized CPS, and all methods in this project were implemented algorithms in publicly available tools (HyST, Hynger, StarL, rtreach, specified in the deliverables). Systems are modeled in Passel as hybrid automata, and the tool generates the CPS semantics in a restricted subclass of first-order logic (FOL) over reals, bitvectors, and integers. Passel leverages recent advances in satisfiability modulo theories (SMT) solvers. Passel exploits the small model theorem we developed to reduce verification for networks composed of arbitrarily many (countably infinite) hybrid automata to checking a network with a (small) finite number [39]. Abstraction results like this enable scalable verification, and allow Passel to automatically prove inductive invariants by checking validity of appropriate FOL formulas. Passel has been applied to verify CPS examples like the Small Aircraft Transportation System (SATS) landing protocol in NASA/FAA NextGen program [41-48].

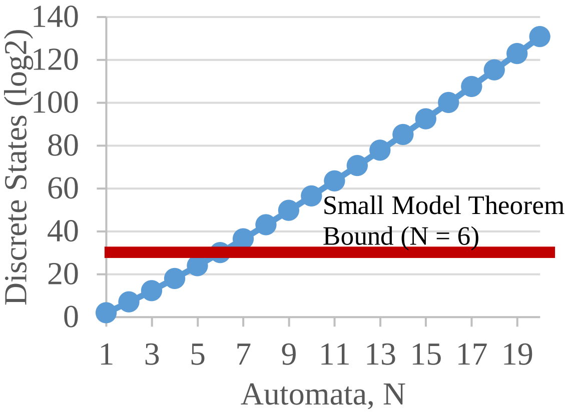


Figure 7: Illustration of the state-space explosion problem for the Small Aircraft Transportation System (SATS) case study modeled as networks of hybrid automata [31, 39, 33], and using a small model theorem to address the problem.

Reductions in Formal Verification: Symmetry-reduction methods [31, 40, 55-63] similarly allow for formally verifying systems with large spaces by only exploring small equivalences classes of the large state space. For example, in preliminary results [31, 40] shown in Figure 8 allow for verification of significantly larger networks of hybrid automata than existing methods (e.g., in PHAVer [76] or SpaceEx [77]). In preliminary results [31, 40] consider systems that have ***on the order of*** $\boldsymbol{2^{130}}$ ***discrete states*** (growing at $N(4N)^N$, see Figure 7) as well as on the ***order of*** $\boldsymbol{N = 20}$ ***to hundreds of continuous variables***, where N is the number of automata in the network (the x-axis in Figure 7 and Figure 8). *No other tool can support such large state spaces with a combination of both complex discrete and continuous behaviors* (e.g., [76-79]) and the closest comparable tool is Uppaal [80] (but that does not support as general dynamics). Leveraging these results, we developed the first formal verification of emergent properties like flocking in DCPS.

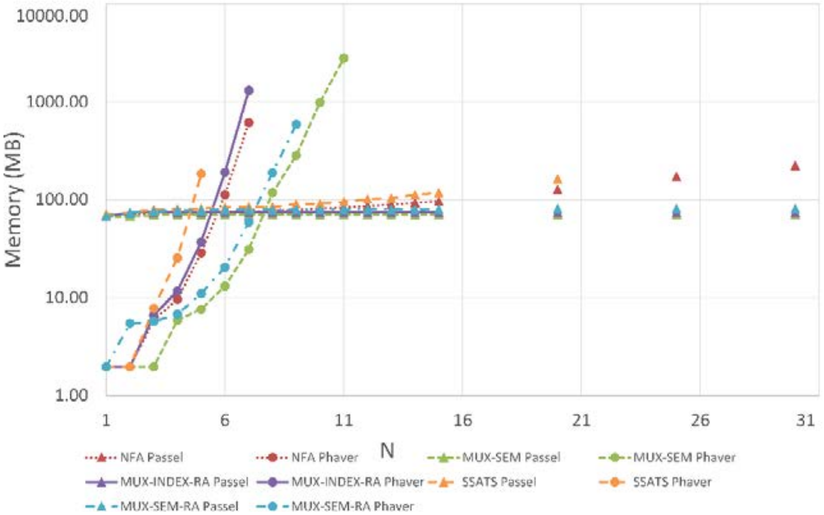


Figure 8: Symmetry-reduced reachability of hybrid automata networks implemented in the Passel verification tool [31, 40, 38], which addresses the state-space explosion problem and allows significantly larger problem size than existing state-of-the-art methodology (in PHAVer).

3.2.2. Detecting Emergence at Runtime: Specification-Based Runtime Monitoring

Objective 2 was the development of runtime monitoring verification methods for emergence in DCPS, using both model-based and model-free approaches. Model-based methods rely on formal methods tools and inherently are subject to scalability problems, while model-free approaches are heuristic, as they are both unsound and incomplete, but scale better. Together, the methods rely on monitoring asynchronous distributed and hybrid systems at runtime and in real-time, and builds upon both model-free and model-based approaches developed by our group [1, 7, 50]. While the model-based design framework and typical formal verification problem assumes a system model $\mathcal{A}$ with formal semantics is available, this is rarely the case in the current state of engineering practice. We developed an invariant synthesis tool called Hynger (HYbrid iNvariant GEneratoR) [50] that instruments arbitrary Simulink/Stateflow (SLSF) block diagrams for input to the Daikon invariant finder [81, 82] to generate candidate invariants over the input and output variables of every block in a diagram. The internals of the SLSF blocks may be unknown, be compiled machine code, actual systems, etc. Such heuristic methods scale better than formal methods alone. However, if the internals are known and formal models are available, the candidates may then be checked to be actual invariants using tools like Passel [31], HyCreate [83], SpaceEx [77], etc., so these heuristic methods enable scalable usage of formal tools for monitoring invariants.

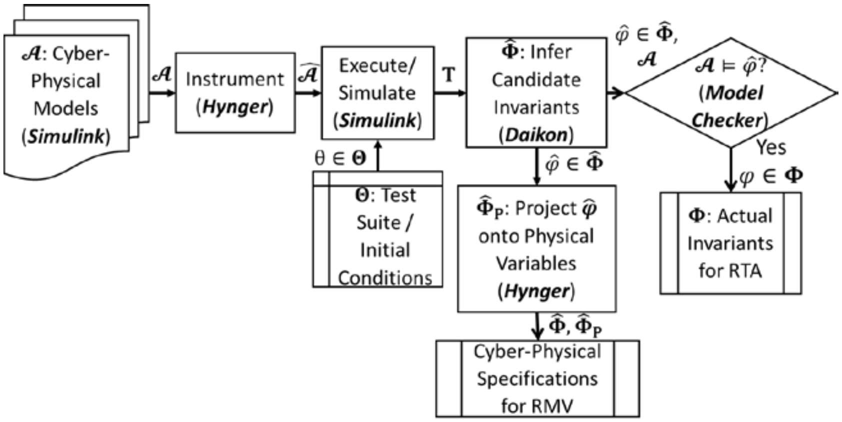

$\mathcal{A}$: Cyber-Physical Models (*Simulink*)
$\mathcal{A}$
Instrument (*Hynger*)
$\widehat{\mathcal{A}}$
Execute/ Simulate (*Simulink*)
T
$\widehat{\Phi}$: Infer Candidate Invariants (*Daikon*)
$\hat{\varphi} \in \widehat{\Phi}$, $\mathcal{A}$
$\mathcal{A} \vDash \hat{\varphi}$? (*Model Checker*)
Yes
$\varphi \in \Phi$
Φ: Actual Invariants for RTA
$\theta \in \Theta$
Θ: Test Suite / Initial Conditions
$\hat{\varphi} \in \widehat{\Phi}$
$\widehat{\Phi}_P$: Project $\hat{\varphi}$ onto Physical Variables (*Hynger*)
$\widehat{\Phi}, \widehat{\Phi}_P$
Cyber-Physical Specifications for RMV

Figure 9: Hynger-based formal and heuristic-based invariant inference for emergent behavior in DCPS. Sets of candidate invariants are generated to either monitor the sets of invariants themselves and how they change over time, as well as prove that these candidates are actual invariants for RTA.

3.2.2.1. Model-Free and Model-Based Invariant Inference and Synthesis for Emergence in DCPS

We first extend the invariant inference methodology to distributed CPS from individual systems currently supported. Combined with emergence specified as invariants, this allows for identifying the presence or absence of emergent behavior in DCPS at runtime. While not all interesting specifications of emergent behavior may be found as invariants, many examples can, such as those for flocking and collision avoidance in (1) and (2).

The overall methodology is depicted in Figure 9. A CPS model or implementation is provided as a SLSF diagram $\mathcal{A}$. The SLSF diagram is instrumented, then the SLSF diagram is executed to generate a set of sampled, finite-precision traces T for each initial condition θ in a set of initial conditions Θ, which effectively corresponds to a test suite. The traces are analyzed using dynamic analysis methods, such as Daikon, to generate a set of **candidate invariants** $\widehat{\Phi}$, each element $\hat{\varphi}$ of which may be checked as actual invariants if $\mathcal{A}$ corresponds to a formal model (e.g., a hybrid automaton), then a model checker may be employed to see if it is an actual invariant φ, and the set of actual invariants Φ is collected. Next, each candidate invariant $\hat{\varphi} \in \widehat{\Phi}$ is projected (restricted) onto the subset of physical variables to yield a candidate physical invariant $\hat{\varphi}_\mathrm{P}$ and corresponding set $\widehat{\Phi}_\mathrm{P}$. *Now, $\widehat{\Phi}_P$ corresponds to the candidate, inferred physical invariants from the perspective of the DCPS*. The candidate sets of invariants and proved invariants are used for runtime monitoring and verification (RMV) and runtime assurance (RTA).

To formalize the problem, an extension of hybrid input/output automata (HIOA) was developed [53, 84-86], called cyber-physical input/output automata (CPIOA) [35]. In addition to partitioning variables into local, input, and output sets, each of these sets of variables are further partitioned into cyber and physical variables. Then, when states (or formulas used to symbolically represent states) are restricted to the set of cyber (respectively, physical) variables, the specifications then correspond to the cyber (respectively, physical) specification. In practical software implementations using e.g., C and SLSF models, the physical variables can be specified using a subtyping of the usual types for approximation of reals (e.g., a physical variable is a subtype of double floating-point or fixed-point types). Techniques building on taint analysis of programs are used to identify the effects of all physical variables in CPS [87].

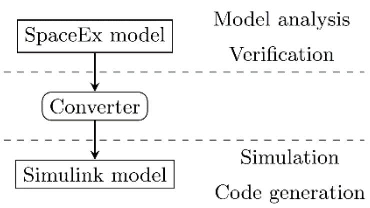


Figure 10: Correct-by-construction implementations of DCPS, starting from a formal model (e.g., as a SpaceEx hybrid automaton network) using a sound translation to implementations as Simulink models.

We utilize both dynamic and static analyses of CPS models to infer the cyber-physical specifications of emergence. When models with formal semantics (e.g., CPIOA) are available, static analysis in the form of reachability analysis may be employed to determine invariant specifications. If no such formal models (or potentially no models and even only black-box implementations are available), one may employ dynamic analysis by executing (or simulating) the systems under consideration to generate sets of executions (or sampled approximate traces, due to inherent inaccuracies of simulation on finite-precision digital computers). We developed a methodology within Hynger for instrumenting arbitrary SLSF diagrams (that may potentially have known or unknown models or system implementations) to generate output traces in the format compatible with the Daikon dynamic invariant inference tool [81, 82]. The SLSF blocks may be unknown models or even system implementations since from the point of view of SLSF, the only information required for blocks are variable values at block inputs and outputs and when that information is updated. For instance, SLSF may be integrated with hardware/software-in-the-loop simulation, and for these purposes, some blocks represent models to be simulated and have information necessary to perform simulation, while other blocks actually correspond to implementations that have been interfaced to provide necessary data to SLSF. Since physical variables evolve according to ODEs, their invariants may involve nonlinear and transcendental functions. Nonlinear (polynomial) invariants [88], disjunctive/max-plus invariants [89, 90], and simulation-based verification (which effectively define invariants from dynamic analysis) [91] may be used to greatly expand the classes of invariants that may be found. If formal models are available, one may check if the inferred invariants are actual invariants using hybrid systems model checkers such as SpaceEx [77], HyCreate [83], and Passel [31, 38]. Physical dynamics and specifications thereof are formalized in a mechanized manner, similar to the numerical simulations formalized in ACSL [92] for Frama-C [93].

Using the formalized distributed emergence inference methods, offline algorithms to identify emergence were developed. As detailed in Figure 9, this results in SMT validity and satisfiability checks over formulas symbolically representing the candidate invariants. We implemented specification inference methods in software tools. A software tool is developed implementing the algorithms developed in the other objectives to solve the emergence inference problem at design time. The software tool is called ***Hynger*** (for Hybrid iNvariant GEneratoR) and integrates with typical CPS development environments (Mathworks Matlab/Simulink) as well as formal analysis tools for hybrid systems, such as Passel [31, 38] and SpaceEx [77]. This leverages extensive experience using SMT solvers [94, 95] such as Z3 [96] used by the Passel tool [31, 38, 40] for the satisfiability/validity checks. Case study models (and the testbed described in Objective 4) are developed to evaluate the inference methods.

We investigated richer specification languages, such as temporal logics (e.g., linear temporal logic [LTL] or computation-tree logic [CTL], etc.), as well as real-time temporal logics (e.g., metric temporal logic [MTL], metric interval temporal logic [MITL], signal temporal logic [STL] [97], etc. [98]). With richer specification languages like LTL, richer techniques are be necessary, and ideas such as Angluin's learning algorithm [99, 100] or counterexample-guided synthesis [97, 101] to infer specifications (i.e., finite-state automata for LTL and parameters for STL) from executions were investigated. The detection of emergence becomes more complex, as instead of satisfiability checks between invariants to determine inclusions, language inclusions must be checked. To work with C code, Daikon must utilize appropriately instrumented binaries using Valgrind via its Kvasir/Fjalar frontends [82]. This makes it difficult to use on non-x86/x86-64 platforms, which is a serious limitation, as most embedded platforms utilize other architectures (e.g., ARM, AVR, PIC, 8051, MSP430, etc.). Due in part to these limitations, the methodology instruments architecture-independent SLSF diagrams to generate traces in the input format compatible with dynamic analysis tools like Daikon. The Hynger tool takes an arbitrary SLSF model, instrument it, then analyze the resulting traces with dynamic analysis to identify broad classes of emergent behavior.

3.2.2.2. Runtime Assurance and Runtime Verification for Emergence in DCPS

Next, the candidate invariants detected using the Hynger and Daikon tools may be monitored at runtime to enable a runtime assurance framework like the ClearView system for distributed (purely software) systems [102, 103]. While technically unsound and incomplete, practically, given a sufficiently large test database, the candidate invariants correspond well to the expected behaviors of the system, and serve as abstractions of all internal behavior. At runtime when analyzing traces over finite times, if the candidate invariants inferred are not implied by known candidates then a suspicious scenario is flagged (such as an attack [102, 103], emergent behavior, etc.). We investigated and use distributed global predicate and state detection algorithms that rely on minimal communication, building upon seminal results of Chandy, Misra, and Lamport [104-106]. Self-stabilization [4] is used as a tool to formalize the emergence specifications and their evolution over time in the distributed systems (as invariants, i.e., predicates of state space).

3.2.3. Assured Control in Spite of Emergence with Real-Time Reachability and Self-Stabilization for Distributed Simplex

Objective 3 is the development of control methods to ensure desirable or avoid undesirable emergent distributed behavior at runtime, by leveraging the Simplex-based RTA framework using real-time reachability of networks of hybrid automata in conjunction with self-stabilization [3-6, 52, 53] of the distributed system. Monitoring predicates over physical variables and their continuous evolution over time is performed with real-time hybrid systems reachability. We developed a methodology for runtime assurance in Simplex-architecture RTA systems using real-time reachability for a single hybrid automaton [7]. These results are restricted to a single hybrid automaton, and require extensions to DCPS. Since modern DCPS are complex, it may be infeasible to determine all specifications and possible emergence between all subcomponents at design time. We developed online runtime monitoring and verification methods for the inferred candidate specifications of emergence, and combine these monitoring methods with real-time algorithms for detecting emergence at runtime. When emergence is identified at runtime, a runtime assurance framework building on supervisory control ensures safe DCPS runtime operation in spite of emergence.

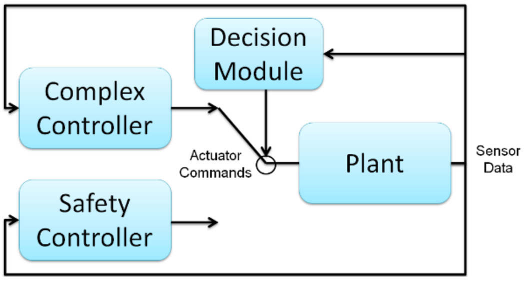


Figure 11: Overview of the Simplex architecture where an unverified, complex controller with verified switching logic (decision module) switches to a verified safety controller in time to prevent mishaps. We extended the architecture to distributed Simplex for DCPS leveraging tools from self-stabilizing distributed systems.

For some of the analysis, we assume formal hybrid automata models are available, which may not be the case for practical CPS that are designed using more typical industrial tools such as Mathworks Simulink/Stateflow (***SLSF***). To alleviate this issue, we investigated a new design paradigm, where the plant, controller, and their interfaces are designed formally as hybrid automata, then are translated to implementations as SLSF diagrams (see Figure 10) [54]. This paradigm of designing with formal models, then instantiating implementations is attractive, as both simulation and verification may be conducted with the formal model, then an implementation may be derived that is guaranteed to have the same behaviors. The sound translation framework from formal hybrid automata models to SLSF diagrams (in particular, continuous-time Stateflow diagrams, which have behaviors similar to hybrid automata) has numerous theoretical and practical challenges. For instance, typical hybrid automata models do not support urgency (although hybrid automata with urgency have been investigated recently [107]), while transitions in SLSF are urgent (i.e., transitions are taken as soon as they are enabled, which is further complicated in SLSF due to actually happening at zero-crossing event points in the simulation loop). SLSF diagrams do not support invariants, while hybrid automata do. SLSF diagrams (without stochastic models) are typically deterministic (in both discrete transitions and continuous trajectories), while hybrid automata are nondeterministic (in both discrete transitions being nondeterministic similar to in nondeterministic finite-state automata [NFAs], and continuous trajectories being described using differential inclusions, which allow for nondeterministic families of solutions). Time-dependent switching is used to abstract more general state-dependent switching. Addressing these issues to ensure a notion of behavior preservation when translating from hybrid automata to SLSF (using an appropriate assumption on the behavior of the SLSF simulation loop and its inherently sampled-time and finite-precision limitations) to enable formal guarantees in implementations.

Next, algorithms were developed for an online, runtime implementation of the overall distributed emergence detection as candidate invariants architecture depicted in Figure 9**.** For the dynamic analysis, the specification inference methodology is implemented online, to infer specifications at runtime. Such methods have been used for identifying security attacks in ClearView [102], but CPS have a different set of challenges (real-time, real value approximations, etc.) [51]. For the static analysis, we built upon preliminary results (Figure 13) for real-time reachability of a single hybrid automaton [7].

The Simplex Architecture (see Figure 11) ensures the safe use of an unverifiable complex controller by using a verified safety controller and verified switching logic [108-113]. This architecture enables the safe use of high-performance, untrusted, and complex control algorithms without requiring them to be formally verified. Simplex incorporates a supervisory controller and safety controller that may take over control if the unverified logic misbehaves. The supervisory controller should guarantee the system never enters an unsafe state (safety), but also use the complex controller as much as possible (minimize conservatism). In preliminary results [7], we establish a combined online/offline approach that uses a real-time reachability computation enables a proof of safety, but with significantly less conservatism, so the upgraded controller is used more frequently as in Figure 13. In this objective, a runtime assurance framework is developed, where the safety controller is used if distributed emergent behavior is detected online.

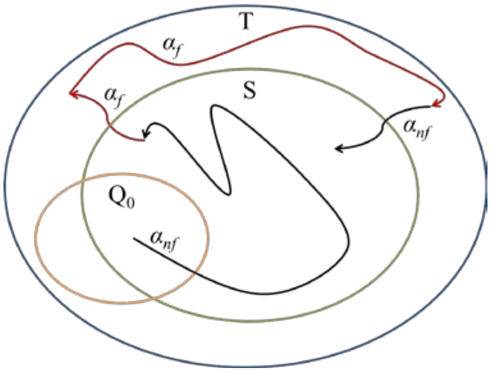


Figure 12: Illustration of self-stabilization. The DCPS starts from a set of initial states Q_0 and if it evolves over an arbitrary execution without failures (attacks, emergent behavior, etc.) α_{nf}, is guaranteed to self-stabilize to a set of desirable states S where mission progress is ensured and remain there. As the DCPS operates, if failures occur and executions α_f are followed outside the set S, they are guaranteed to remain in a set T that at least maintains safety. Once failures stop, the DCPS again self-stabilizes to S and may make mission progress.

Hynger has been extended for runtime assurance tasks like detecting and thwarting security violations and attacks, similar to the ClearView tool that also relies on dynamic analysis to detect changes in candidate specifications [51, 102]. Finding and monitoring sets of candidate invariants (even if not verified as actual invariants) may be useful for runtime assurance and resiliency methods for embedded systems. If candidate invariants are checked at runtime using a real-time reachability method [7], formal and dynamic runtime assurance may be feasible. Rather than purely sensing feedback in the Simplex decision, using changes in sets of inferred candidate invariants may determine mode changes to enable runtime assurance in DCPS.

3.2.3.1. Real-Time Reachability for Networks of Hybrid Automata

The next research objective is to extend real-time reachability to DCPS modeled as networks of hybrid automata, which is the first step in developing an RTA framework for DCPS. Existing methods have only been developed for a single hybrid automaton, so the focus is on developing a ***distributed*** runtime verification method building on the real-time reachability of networks of hybrid automata. This is enabled by extending the symmetry-reducing reachability framework for networks of hybrid automata [40] to those with linear and nonlinear dynamics and specifications, developed in Objective 1.

3.2.3.2. RTA for Emergence in DCPS with Distributed Simplex and Self-Stabilization

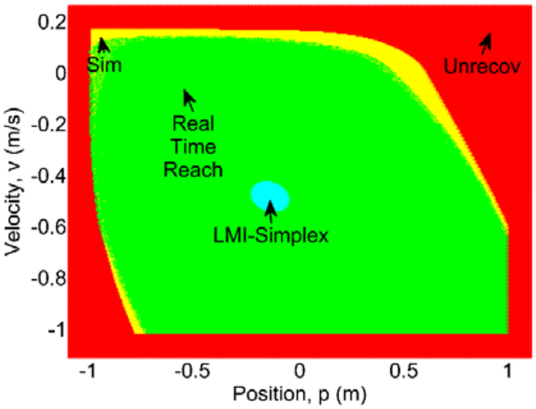


Figure 13: Verifiably safe regions of state-space when using complex controller for an inverted pendulum example, illustrating real-time reachability's advantages to offline verification methods (unverified simulation or LMI-based methods that yield ellipsoidal safe sets) in results of a Simplex RTA framework.

Leveraging both the real-time reachability for hybrid automata networks and the Hynger-based emergence monitoring methods from Objective 2, the next objective is to apply these monitoring methods in RTA control of emergence, which is specified as maintaining system state within a given region of the state-space (i.e., property invariance). We build on the theory and tools of self-stabilizing distributed systems (see Figure 12), which ensures eventually returning to desirable sets of states in spite of failures, attacks, etc. Together with the Simplex RTA methods, thus yields the development of a Distributed Simplex RTA architecture for DCPS. This combines global and local state estimation and invariant monitoring. For example, each agent may deploy its own Simplex architecture, but what emergent behaviors occur if say all agents start to use fallback controllers? What is a fallback controller for the entire distributed system?

These questions have been addressed in our resulting publications. We leverage preliminary results [5-7, 52, 53, 114] in this direction to develop a Simplex architecture for emergence in DCPS.

3.2.4. Evaluating Analysis, Monitoring, and Control of Emergence in DCPS Testbeds

To evaluate the methods from the previous objectives, we performed simulations, laboratory experiments, and demonstrations using a swarm of autonomous agents, particularly commercially available quadrotor drones. Typical safety properties that arise are collision avoidance [5, 6, 31-33, 39, 52] and convergence to some desired configuration and/or location [52], and emergent properties may be consensus, flocking, or unwanted oscillatory movements due to failures, attacks, communication delays, etc. Our results in similar studies include verification of autonomous satellite maneuvers [2], flocking in swarm robotics in spite of failures [5, 6], and planar robotics [52].

3.2.4.1. Evaluation of Emergence Methods through Simulation Studies

We evaluated the specification, verification, monitoring, and control methods analytically, using software tools, and in simulation. We extended the StarL framework that provides simulation capability of DCPS. Additionally, StarL was used to deploy to actual swarm robot systems, so altogether this enables evaluation of a correct-by-construction framework for establishing or avoiding emergence in DCPS. The StarL [114, 115] platform and its offline simulator allows the DCPS to have similar levels of concurrency, asynchrony, and other realistic effects as implementations. We used the hybrid automaton translation framework (Figure 10) to convert from formal models to StarL programs and SLSF diagrams for simulation.

3.2.4.1: Experimental Evaluation of Emergence Methods through Lab Demonstrations

We experimentally analyzed the specification, verification, monitoring, and control methods for emergent behavior in DCPS using an indoor swarm robotics system of quadrotors. This includes scenarios with emergent behavior such as flocking, flocking in spite of failures of physical, cyber, and communication components, and emergent behavior like collision avoidance that should be invariant. This serves to validate the analytical, verification, and simulation results, and leverages the implementation of StarL programs [114, 115] on hardware.

4. Results and Discussion

4.1. Key Results and Findings

The key results and findings of this project for each objective are as follows.

Objective 1: Specification and Verification

The first is in specifying behaviors for DCPS, and this resulted in the creation of a novel formal specification language called hyperproperties for signal temporal logic (HyperSTL), which arguably is the most complete specification language for formally describing behaviors of DCPS [36]. Also for specification of behavior, the perspective of considering cyber, physical, and cyber-physical specifications in DCPS is a key insight [35]. The second is in addressing the state-space explosion problem for DCPS, particularly through the use of order-reduction [116].

Objective 2: Monitoring

For monitoring DCPS behavior, the runtime monitoring framework built using Hynger to check if behaviors observed at runtime is the key result. By monitoring whether specifications may be violated at runtime gives an indication that emergent behavior, or some other anomalous behavior, may be occurring [51].

Objective 3: Control

The perspective of runtime assurance using the Simplex architecture seems particularly powerful for mitigating emergent behavior, if it is undesirable [117]. This approach may be impactful and useful when artificial intelligence (AI) and machine learning (ML) components are incorporated in DCPS, and this direction of research was a key outcome.

Objective 4: Evaluation

The methods above were evaluated in several case studies, particularly within the distributed robotics framework of StarL. Videos of scenarios are included with the deliverables. Numerous benchmark case studies were published [118-122].

4.2. Related Work

Model-based verification typically develops a model of a system and properties (specifications) are (manually, semi-automatically, or automatically) checked for that model. However, most safety issues induced by software bugs are not a result of design errors, but are the result of implementation, reuse, upgrade, and maintenance errors. While a priori model-based design (MBD) and clean-slate approaches like DARPA's HACMS, seL4, Bedrock, and CompCert [123-127] are of critical important and especially useful for subcomponent verification, most systems being designed today utilize a development process where engineers write software and systems are integrated from numerous components. Additionally, while there are many standards to help improve CPS safety in various domains (like ISO 26262 functional safety standard for road vehicles [128] and MISRA C [129]), as CPS have exponential gains in software embedded in them, these reliability problems will only become exacerbated [8, 9]. Rare cyber-physical failure scenarios and distributed emergence motivate runtime contingencies to assure safe, if degraded, operation.

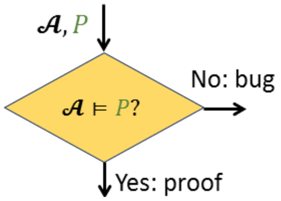


Figure 14: General formal verification problem.

Dynamic Specification Inference: There are many benefits of *dynamic analysis* such as using implementations instead of models [81, 82, 130] to find dynamic program specifications [130]. The limitation is results are unsound without additional reasoning. Finding specifications of systems is a maturing field within software engineering [81, 82, 130-133], and recent simulation-based approaches in hybrid systems and CPS like those used in S-TaLiRo, Breach, and C2E2 can be viewed as dynamic analysis [91, 97, 134-139]. Invariants are properties of a system that always hold, while conditional invariants may hold at certain program points, for example, at the beginning or end of a function call (pre/post conditions). Daikon has found candidate invariants of hybrid models of biological system [140] and distributed systems [141, 142], and this illustrates a proof-of-concept to use it for hybrid systems. Alternative approaches analyze simulation traces from complex Matlab/Simulink models [97, 114, 138, 139], but require a priori specifications or require templates from restricted classes of logic.

Verification of Hybrid Systems: Formal verification aims to solve the problem posed in Figure 14: does a given formal system model (often an automaton) $\mathcal{A}$ satisfy a given specification (property) P? Automated formal verification (as instantiated, for example, in model checkers), aims to develop an algorithm to solve the formal verification problem, instead of using semi-automated methods such as interactive theorem provers. A ***hybrid automaton*** [30, 31, 84, 143, 144] is a formal model, and is essentially a finite-state machines with additional continuous variables that may evolve according to ordinary differential equations (ODEs) or inclusions that may differ in each state. Hybrid automata provide a formal mathematical semantics for formal verification of properties specified in some formal language using many techniques [31, 76-78, 144]. While automated formal verification is undecidable for many interesting classes of systems (such as general software or general hybrid automata), numerous advances have been made in the past few decades. Explicit-state and symbolic model checking [68, 145-149] is common place in numerous industrial semiconductor development processes, aided in part by automata-theoretic advances and developments like efficient representations like BDDs [150], DDDs [151], SAT-encodings [152, 153], and recently, SMT-encodings [94, 95, 154-157] and quantified encodings [158]. Embedded and hybrid systems have likewise benefited from advances such as those implemented in HyTech [78], KRONOS [159], Charon [160], Checkmate [161], the ellipsoidal toolbox [162], PHAVer [76], KeYmaera [163, 164], SpaceEx [77, 165], and simulation-based verification [134-136, 139, 166-168]. SMT-based techniques have been used for reachability analysis of hybrid systems in SAT-modulo-ODEs [79, 169-171], and for automatically discharging deductive proofs of safety by inductive invariance in Passel [31, 32, 38-40].

Translating Hybrid Automata to Implementations: Efforts have recently been investigated for translating timed automata to Mathworks Simulink/Stateflow (***SLSF***) diagrams, such as UPP2SF that converts UPPAAL's timed automata to SLSF diagrams while maintaining certain properties of executions (i.e., a form of soundness) [172-175]. A vast amount of existing work exists in the opposite direction, of translating from SLSF to formal models like hybrid automata, extended finite state machines (EFSMs), etc. and between hybrid systems formalisms [176-187]. However, tools translating from SLSF are impractically difficult to build, in part since SLSF does not have a formal semantics (although efforts have tried to define some [188, 189]), and commercially available tools such as Ansys/Esterel's SCADE Lustre [190] converter tool (that translates SLSF to Lustre with precise semantics) are not only impracticably large to build in an academic setting (e.g., Esterel's converter has millions of lines of code), but are theoretically unsound, albeit very useful. Additionally, all commercially viable converters only support discrete SLSF diagrams (or discretizations thereof), and may not include continuous-time blocks like continuous-time Stateflow diagrams [183, 185, 191]. Academic efforts exist to translate from SLSF to hybrid models (such as HyLink [192-194] and others [176, 184]), but the vast effort in creating viable translators make it impractical.

4.3. Deliverables

Table 1 describes the deliverables produced through this project, which includes quarterly status reports, final technical reports, software deliverables including source code and prototypes, and APIs including documentation.

Objective	**Deliverables**
Objective 1: Modeling and Analysis	HyST/Passel software tool, with extensions to linear and nonlinear local dynamics allowing modeling of the swarm robotics DCPS case study and emergent properties in general DCPS. Software deliverables with source code and prototypes, and APIs including documentation. Online: https://github.com/verivital/hyst
Objective 2: Monitoring	Hynger invariant inference software tool for distributed emergence monitoring. Software deliverables with source code and prototypes, and APIs including documentation. Online: https://bitbucket.org/verivital/hynger

Objective 3: Control	Real-time reachability tool and RTA framework for emergence in DCPS relying on self-stabilization. Software deliverables with source code and prototypes, and APIs including documentation. Online: https://bitbucket.org/verivital/rtreach
Objective 4: Evaluation	Models of the swarm robot case studies; source code for the control software; source code and design files for the overall swarm robot evaluation system. Software deliverables with source code and prototypes, and APIs including documentation. Online: https://github.com/verivital/starl Videos: https://www.youtube.com/channel/UC1-RPjoacWVNLQKjuxrbn9A

Table 1: Deliverables for analysis, monitoring, and control of emergence in DCPS.

5. Conclusion

This project studied emergent behavior in DCPS by developing formal specification languages, formal verification methods within the HyST software tool, heuristic-based runtime monitoring within the Hynger software tool, and Simplex-based runtime assurance. Together, the project demonstrates the capability to detect, monitor, and control emergent behavior in DCPS.

6. References

[1] L. Bobadilla, T. T. Johnson, and A. LaViers, “Verified planar formation control algorithms by composition of primitives,” in *AIAA SciTech*. Kissimmee, Florida: AIAA, Jan. 2015.

[2] T. T. Johnson, J. Green, S. Mitra, R. Dudley, and R. S. Erwin, “Satellite rendezvous and conjunction avoidance: Case studies in verification of nonlinear hybrid systems,” in *Proceedings of the 18th International Conference on Formal Methods (FM 2012)*, D. Giannakopoulou and D. Méry, Eds. Paris, France: Springer Berlin Heidelberg, Aug. 2012, vol. 7436, pp. 252–266.

[3] A. Arora and M. Gouda, “Closure and convergence: A foundation of fault-tolerant computing,” vol. 19, pp. 1015–1027, 1993.

[4] S. Dolev, *Self-stabilization*. Cambridge, MA: MIT Press, 2000.

[5] T. T. Johnson and S. Mitra, “Safe flocking in spite of actuator faults using directional failure detectors,” *Journal of Nonlinear Systems and Applications*, vol. 2, no. 1-2, pp. 73–95, Apr. 2011.

[6] ——, “Safe flocking in spite of actuator faults,” in *12th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2010)*, ser. Lecture Notes in Computer Science, S. Dolev, J. Cobb, M. Fischer, and M. Yung, Eds. Springer Berlin / Heidelberg, Sep. 2010, vol. 6366, pp. 588–602.

[7] S. Bak, T. T. Johnson, M. Caccamo, and L. Sha, “Real-time reachability for verified simplex design,” in *IEEE Real-Time Systems Symposium (RTSS)*. Rome, Italy: IEEE Computer Society, Dec. 2014.

[8] M. Broy, “Challenges in automotive software engineering,” in *Proceedings of the 28th International Conference on Software Engineering*, ser. ICSE ’06. New York, NY, USA: ACM, 2006, pp. 33–42.

[9] M. Broy, I. Kruger, A. Pretschner, and C. Salzmann, “Engineering automotive software,” *Proceedings of the IEEE*, vol. 95, no. 2, pp. 356–373, Feb. 2007.

[10] R. N. Charette, “This car runs on code,” *IEEE Spectrum*, 2009.

[11] B. Beizer, *Software testing techniques (2nd ed.)*. New York, NY, USA: Van Nostrand Reinhold Co., 1990.

[12] G. Tassey, “The economic impacts of inadequate infrastructure for software test,” National Institute of Standards and Technology, Tech. Rep. Planning Report 02-3, May 2002.

[13] Consumer Product Safety Commission, “Fire alarm control panels recalled by fire-lite alarms due to alert failure (alert #11-702),” Oct. 2010. [Online]. Available: http://www.cpsc.gov/-en/Recalls/2011/Fire-Alarm-Control-Panels-Recalled-by-Fire-Lite-Alarms-Due-to-Alert-Failure/

[14] ——, “Simplex fire alarm control panels recalled by tyco safety products westminster due to failure to alert monitoring centers (alert #11-721),” Feb. 2011. [Online]. Available: http://-www.cpsc.gov/en/Recalls/2011/Simplex-Fire-Alarm-Control-Panels-Recalled-by-Tyco-Safety-Products-Westminster-Due-to-Failure-to-Alert-Monitoring-Centers/

[15] ——, “Fire control panels recalled by bosch security systems corp. due to alarm failure posing a fire hazard (alert #12-721),” Feb. 2012. [Online]. Available: http://www.cpsc.gov/en/-Recalls/2012/Fire-Control-Panels-Recalled-by-Bosch-Security-Systems-Corp-Due-to-Alarm-Failure-Posing-a-Fire-Hazard/

[16] H. Alemzadeh, R. Iyer, Z. Kalbarczyk, and J. Raman, “Analysis of safety-critical computer failures in medical devices,” *Security Privacy, IEEE*, vol. 11, no. 4, pp. 14–26, 2013.

[17] “Fda medical device recall report 2003 to 2012,” Food and Drug Administration, Tech. Rep., Mar. 2014. [Online]. Available: http://www.fda.gov/downloads/AboutFDA/-CentersOffices/OfficeofMedicalProductsandTobacco/CDRH/CDRHTransparency/-UCM388442.pdf

[18] National Highway Traffic Safety Administration (NHTSA), “(action #pe05029),” Oct. 2005. [Online]. Available: http://www-odi.nhtsa.dot.gov/cars/problems/defect/-results.cfm?action_number=PE05029&SearchType=QuickSearch&summary=true

[19] A. Saadat, “Defect information report (NHTSA Recall 14V-053),” Feb. 2014. [Online]. Available: http://www-odi.nhtsa.dot.gov/acms/cs/jaxrs/download/doc/UCM450071/RCDNN-14V053-0945.pdf

[20] National Highway Traffic Safety Administration (NHTSA), “Honda automatic transmission control module software (recall #11v395000),” Aug. 2011.

[21] E. Shaw, “Fish in schools,” *Natural History*, vol. 84, no. 8, pp. 40–45, 1975.

[22] C. W. Reynolds, “Flocks, herds and schools: A distributed behavioral model,” in *SIGGRAPH ’87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*. New York, NY, USA: ACM, 1987, pp. 25–34.

[23] C. F. Daganzo, “The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory,” *Transportation Research Part B: Methodological*, vol. 28, no. 4, pp. 269 – 287, 1994.

[24] W. Jones, “Forecasting traffic flow,” *Spectrum, IEEE*, vol. 38, no. 1, pp. 90–91, 2001.

[25] R. Olfati-Saber and R. Murray, “Consensus problems in networks of agents with switching topology and time-delays,” vol. 49, no. 9, pp. 1520–1533, Sep. 2004.

[26] R. Olfati-Saber, “Flocking for multi-agent dynamic systems: algorithms and theory,” vol. 51, no. 3, pp. 401–420, Mar. 2006.

[27] R. Olfati-Saber, J. Fax, and R. Murray, “Consensus and cooperation in networked multi-agent systems,” *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.

[28] T. D. Chandra and S. Toueg, “Unreliable failure detectors for reliable distributed systems,” *J. ACM*, vol. 43, pp. 225–267, Mar. 1996.

[29] M. J. Fischer, N. A. Lynch, and M. S. Paterson, “Impossibility of distributed consensus with one faulty process,” *J. ACM*, vol. 32, no. 2, pp. 374–382, 1985.

[30] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, “The algorithmic analysis of hybrid systems,” *Theoretical Computer Science*, vol. 138, no. 1, pp. 3–34, 1995.

[31] T. T. Johnson, “Uniform verification of safety for parameterized networks of hybrid automata,” Ph.D. dissertation, University of Illinois at Urbana-Champaign, Electrical and Computer Engineering, Urbana, IL 61801, 2013.

[32] T. T. Johnson and S. Mitra, “Invariant synthesis for verification of parameterized cyber-physical systems with applications to aerospace systems,” in *Proceedings of the AIAA Infotech at Aerospace Conference (AIAA Infotech 2013)*, Boston, MA, Aug. 2013.

[33] ——, “Parameterized verification of distributed cyber-physical systems: An aircraft landing protocol case study,” in *ACM/IEEE 3rd International Conference on Cyber-Physical Systems*, Apr. 2012.

[34] L. V. Nguyen and T. T. Johnson, “Virtual prototyping and distributed control for solar array with distributed multilevel inverter,” *CoRR*, vol. abs/1404.2259, 2014.

[35] L. V. Nguyen, K. Hoque, S. Bak, S. Drager, and T. T. Johnson, “Cyber-physical specification mismatches,” *ACM Transactions on Cyber-Physical Systems (<a href="http://www.acm.org/TCPS/">TCPS</a>)*, 2018.

[36] L. V. Nguyen, J. Kapinski, X. Jin, J. Deshmukh, and T. T. Johnson, “Hyperproperties of real-valued signals,” in *15th ACM-IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE 2017)*. IEEE, Oct. 2017.

[37] L. V. Nguyen, J. Kapinski, X. Jin, J. Deshmukh, K. Butts, and T. T. Johnson, “Abnormal data classification using time-frequency temporal logic,” in *20th Intl. Conf. on Hybrid Systems: Computation and Control (HSCC 2017)*. ACM, Apr. 2017.

[38] [Online]. Available: http://publish.illinois.edu/passel-tool/

[39] T. T. Johnson and S. Mitra, “A small model theorem for rectangular hybrid automata networks,” in *Proceedings of the IFIP International Conference on Formal Techniques for Distributed Systems, Joint 14th Formal Methods for Open Object-Based Distributed Systems and 32nd Formal Techniques for Networked and Distributed Systems (FMOODS-FORTE)*, ser. LNCS. Springer, June 2012, vol. 7273.

[40] ——, “Anonymized reachability of rectangular hybrid automata networks,” in *Formal Modeling and Analysis of Timed Systems (FORMATS)*, 2014.

[41] T. S. Abbott, M. C. Consiglio, B. T. Baxley, D. M. Williams, K. M. Jones, and C. A. Adams, “Small aircraft transportation system higher volume operations concept,” NASA, Tech. Rep. NASA/TP-2006-214512, L-19215, Oct. 2006.

[42] T. S. Abbott, K. M. Jones, M. C. Consiglio, D. M. Williams, and C. A. Adams, “Small aircraft transportation system, higher volume operations concept: Normal operations,” NASA, Tech. Rep. NASA/TM-2004-213022, Aug. 2004.

[43] C. A. Adams, J. L. Murdoch, M. C. Consiglio, and D. M. Williams, “Incorporating data link messaging into a multi-function display to support the small aircraft transportation system (sats) and the self-separation of general aviation aircraft,” *Applied Ergonomics*, vol. 38, no. 4, pp. 465–471, 2007.

[44] V. Carreño and C. Muñoz, “Safety verification of the small aircraft transportation system concept of operations,” in *Proceedings of the AIAA 5th Aviation, Technology, Integration, and Operations Conference, AIAA-2005-7423*, Arlington, Virginia, 2005.

[45] C. Muñoz, V. Carreño, and G. Dowek, “Formal analysis of the operational concept for the small aircraft transportation system,” in *Rigorous Development of Complex Fault-Tolerant Systems*, ser. LNCS, M. Butler, C. Jones, A. Romanovsky, and E. Troubitsyna, Eds. Springer, 2006, vol. 4157, pp. 306–325.

[46] S. Umeno and N. Lynch, “Proving safety properties of an aircraft landing protocol using I/O automata and the PVS theorem prover: A case study,” in *Formal Methods*, ser. LNCS, J. Misra, T. Nipkow, and E. Sekerinski, Eds. Springer, 2006, vol. 4085, pp. 64–80.

[47] ——, “Safety verification of an aircraft landing protocol: A refinement approach,” in *Hybrid Systems: Computation and Control*, ser. LNCS. Springer, 2007, vol. 4416, pp. 557–572.

[48] S. Viken and F. Brooks, “Demonstration of four operating capabilities to enable a small aircraft transportation system,” in *The 24th Digital Avionics Systems Conference (DASC 2005)*, vol. 2, Oct. 2005.

[49] S. Bak, S. Bogomolov, and T. T. Johnson, “HyST: A source transformation and translation tool for hybrid automaton models,” in *Proc. of the 18th Intl. Conf. on Hybrid Systems: Computation and Control (HSCC)*. ACM, 2015.

[50] T. T. Johnson, S. Bak, and S. Drager, “Cyber-physical specification mismatch identification with dynamic analysis,” in *International Conference on Cyber-Physical Systems (ICCPS)*, 2015. [Online]. Available: http://verivital.uta.edu/hynger

[51] O. A. Beg, T. T. Johnson, and A. Davoudi, “Detection of false-data injection attacks in cyber-physical dc microgrids,” *IEEE Transactions on Industrial Informatics*, 2017.

[52] T. T. Johnson, S. Mitra, and K. Manamcheri, “Safe and stabilizing distributed cellular flows,” in *Proceedings of the 30th IEEE International Conference on Distributed Computing Systems (ICDCS)*. Genoa, Italy: IEEE, June 2010, pp. 577–586.

[53] T. T. Johnson, “Fault-tolerant distributed cyber-physical systems: Two case studies,” Master’s thesis, Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, IL 61801, May 2010.

[54] S. Bak, O. A. Beg, S. Bogomolov, T. T. Johnson, L. V. Nguyen, and C. Schilling, “Hybrid automata: from verification to implementation,” *Software Tools for Technology Transfer (STTT)*, Aug. 2017.

[55] E. A. Emerson and A. P. Sistla, “Symmetry and model checking,” *Formal Methods in System Design*, vol. 9, no. 1-2, pp. 105–131, 1996.

[56] C. N. Ip and D. L. Dill, “Better verification through symmetry,” *Formal Methods in System Design*, vol. 9, pp. 41–75, 1996.

[57] D. Tang, S. Malik, A. Gupta, and C. Ip, “Symmetry reduction in sat-based model checking,” in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, K. Etessami and S. Rajamani, Eds. Springer Berlin Heidelberg, 2005, vol. 3576, pp. 125–138.

[58] E. M. Clarke, R. Enders, T. Filkorn, and S. Jha, “Exploiting symmetry in temporal logic model checking,” *Formal Methods in System Design*, vol. 9, pp. 77–104, 1996.

[59] W. D. Obal and W. H. Sanders, “Measure-adaptive state-space construction,” *Performance Evaluation*, vol. 44, no. 1–4, pp. 237–258, 2001.

[60] M. Hendriks, G. Behrmann, K. G. Larsen, P. Niebert, and F. W. Vaandrager, “Adding symmetry reduction to UPPAAL,” in *Formal Modeling and Analysis of Timed Systems*

(FORMATS ’03), ser. LNCS, K. G. Larsen and P. Niebert, Eds., no. 2791. Springer–Verlag, 2004, pp. 46–59.

[61] E. Emerson and T. Wahl, “Dynamic symmetry reduction,” in *Tools and Algorithms for the Construction and Analysis of Systems*, ser. LNCS, N. Halbwachs and L. Zuck, Eds. Springer, 2005, vol. 3440, pp. 382–396.

[62] W. D. Obal, M. McQuinn, and W. Sanders, “Detecting and exploiting symmetry in discrete-state Markov models,” *Reliability, IEEE Transactions on*, vol. 56, no. 4, pp. 643–654, Dec. 2007.

[63] T. Wahl, N. Blanc, and E. Emerson, “SVISS: Symbolic verification of symmetric systems,” in *Tools and Algorithms for the Construction and Analysis of Systems*, ser. LNCS, C. Ramakrishnan and J. Rehof, Eds. Springer, 2008, vol. 4963, pp. 459–462.

[64] E. M. Clarke and O. Grumberg, “Avoiding the state explosion problem in temporal logic model checking,” in *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, ser. PODC ’87. New York, NY, USA: ACM, 1987, pp. 294–303. [Online]. Available: http://doi.acm.org/10.1145/41840.41865

[65] K. McMillan, “Using unfoldings to avoid the state explosion problem in the verification of asynchronous circuits,” in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, G. von Bochmann and D. Probst, Eds. Springer Berlin / Heidelberg, 1993, vol. 663, pp. 164–177.

[66] H. Ejersbo Jensen, K. Guldstrand Larsen, and A. Skou, “Scaling up UPPAAL,” in *Formal Techniques in Real-Time and Fault-Tolerant Systems*, ser. Lecture Notes in Computer Science, M. Joseph, Ed. Springer Berlin / Heidelberg, 2000, vol. 1926, pp. 641–678.

[67] K. S. Namjoshi, “Model Checking in Bits and Pieces,” *ArXiv e-prints*, Sep. 2013.

[68] E. M. Clarke, O. Grumberg, and D. Peled, *Model Checking*. MIT Press, 1999.

[69] A. Pnueli, S. Ruah, and L. Zuck, “Automatic deductive verification with invisible invariants,” in *Tools and Algorithms for the Construction and Analysis of Systems*, ser. LNCS. Springer, 2001, vol. 2031, pp. 82–97.

[70] T. Arons, A. Pnueli, S. Ruah, Y. Xu, and L. Zuck, “Parameterized verification with automatically computed inductive assertions?” in *Computer Aided Verification*, ser. LNCS, G. Berry, H. Comon, and A. Finkel, Eds. Springer, 2001, vol. 2102, pp. 221–234.

[71] A. Pnueli, Y. Rodeh, O. Strichman, and M. Siegel, “The small model property: How small can it be?” *Information and Computation*, vol. 178, no. 1, pp. 279–293, 2002.

[72] L. Zuck and A. Pnueli, “Model checking and abstraction to the aid of parameterized systems,” *Computer Languages, Systems, and Structures*, vol. 30, no. 3-4, pp. 139–169, 2004.

[73] Y. Fang, N. Piterman, A. Pnueli, and L. Zuck, “Liveness with incomprehensible ranking,” in *Tools and Algorithms for the Construction and Analysis of Systems*, ser. LNCS, K. Jensen and A. Podelski, Eds. Springer, 2004, vol. 2988, pp. 482–496.

[74] ——, “Liveness with invisible ranking,” *International Journal on Software Tools for Technology Transfer (STTT)*, vol. 8, pp. 261–279, 2006.

[75] K. Namjoshi, “Symmetry and completeness in the analysis of parameterized systems,” in *Verification, Model Checking, and Abstract Interpretation*, ser. Lecture Notes in Computer Science, B. Cook and A. Podelski, Eds. Springer Berlin / Heidelberg, 2007, vol. 4349, pp. 299–313.

[76] G. Frehse, “PHAVer: Algorithmic verification of hybrid systems past HyTech,” *International Journal on Software Tools for Technology Transfer (STTT)*, vol. 10, pp. 263–279, 2008.

[77] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, “SpaceEx: Scalable verification of hybrid systems,” in *Computer Aided Verification (CAV)*, ser. LNCS. Springer, 2011.

[78] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi, “HyTech: A model checker for hybrid systems,” *Journal on Software Tools for Technology Transfer*, vol. 1, pp. 110–122, 1997.

[79] S. Gao, S. Kong, and E. Clarke, “Satisfiability modulo ODEs,” in *International Conference on Formal Methods in Computer-Aided Design (FMCAD)*, Oct. 2013.

[80] J. Bengtsson, K. Larsen, F. Larsson, P. Pettersson, and W. Yi, “UPPAAL: A tool suite for automatic verification of real-time systems,” in *Hybrid Systems III*, ser. LNCS, R. Alur, T. Henzinger, and E. Sontag, Eds. Springer, 1996, vol. 1066, pp. 232–243.

[81] M. Ernst, J. Cockrell, W. G. Griswold, and D. Notkin, “Dynamically discovering likely program invariants to support program evolution,” *Software Engineering, IEEE Transactions on*, vol. 27, no. 2, pp. 99–123, 2001.

[82] M. D. Ernst, J. H. Perkins, P. J. Guo, S. McCamant, C. Pacheco, M. S. Tschantz, and C. Xiao, “The Daikon system for dynamic detection of likely invariants,” *Science of Computer Programming*, vol. 69, no. 1–3, pp. 35–45, Dec. 2007.

[83] S. Bak. (2013) HyCreate: A tool for overapproximating reachability of hybrid automata. [Online]. Available: http://stanleybak.com/projects/hycreate/hycreate.html

[84] N. Lynch, R. Segala, and F. Vaandrager, “Hybrid I/O automata,” *Information and Computation*, vol. 185, no. 1, pp. 105–157, 2003.

[85] N. Lynch and M. Tuttle, “An introduction to Input/Output automata,” *CWI-Quarterly*, vol. 2, no. 3, pp. 219–246, Sep. 1989.

[86] D. K. Kaynar, N. Lynch, R. Segala, and F. Vaandrager, *The Theory of Timed I/O Automata*, ser. Synthesis Lectures in Computer Science. Morgan & Claypool, 2006.

[87] E. Schwartz, T. Avgerinos, and D. Brumley, “All you ever wanted to know about dynamic taint analysis and forward symbolic execution (but might have been afraid to ask),” in *Security and Privacy (SP), 2010 IEEE Symposium on*, May 2010, pp. 317–331.

[88] T. Nguyen, D. Kapur, W. Weimer, and S. Forrest, “Using dynamic analysis to discover polynomial and array invariants,” in *Proceedings of the 34th International Conference on Software Engineering*, ser. ICSE ’12. Piscataway, NJ, USA: IEEE Press, 2012, pp. 683–693.

[89] ——, “Using dynamic analysis to generate disjunctive invariants,” in *Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE 2014. New York, NY, USA: ACM, 2014, pp. 608–619.

[90] ——, “DIG: A dynamic invariant generator for polynomial and array invariants,” *ACM Transactions on Software Engineering and Methodology, to appear*, 2014.

[91] Z. Huang and S. Mitra, “Proofs from simulations and modular annotations,” in *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control*, ser. HSCC ’14. New York, NY, USA: ACM, 2014, pp. 183–192.

[92] *ANSI C Specification Language (ACSL)*, ANSI Std.

[93] S. Boldo, F. Clément, J.-C. Filliâtre, M. Mayero, G. Melquiond, and P. Weis, “Wave equation numerical resolution: A comprehensive mechanized proof of a c program,” *Journal of Automated Reasoning*, vol. 50, no. 4, pp. 423–456, 2013.

[94] L. De Moura and N. Bjørner, “Satisfiability modulo theories: Introduction and applications,” *Commun. ACM*, vol. 54, pp. 69–77, Sep. 2011.

[95] C. Barrett and C. Tinelli, “Satisfiability modulo theories,” in *Handbook of Model Checking*, E. Clarke, T. Henzinger, and H. Veith, Eds., 2014.

[96] L. De Moura and N. Bjørner, “Z3: An efficient SMT solver,” in *Proc. of 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, ser. TACAS ’08/ETAPS ’08. Springer-Verlag, 2008, pp. 337–340.

[97] X. Jin, A. Donzé, J. V. Deshmukh, and S. A. Seshia, “Mining requirements from closed-loop control models,” in *Proceedings of the 16th international conference on Hybrid systems: computation and control*, ser. HSCC ’13. New York, NY, USA: ACM, 2013, pp. 43–52.

[98] J. Ouaknine and J. Worrell, “Some recent results in metric temporal logic,” in *Formal Modeling and Analysis of Timed Systems*, ser. LNCS, F. Cassez and C. Jard, Eds. Springer Berlin Heidelberg, 2008, vol. 5215, pp. 1–13.

[99] D. Angluin, “Learning regular sets from queries and counterexamples,” *Information and Computation*, vol. 75, no. 2, pp. 87 – 106, 1987.

[100] T. Berg, B. Jonsson, M. Leucker, and M. Saksena, “Insights to angluin’s learning,” *Electronic Notes in Theoretical Computer Science*, vol. 118, no. 0, pp. 3 – 18, 2005, proceedings of the International Workshop on Software Verification and Validation (SVV 2003) Software Verification and Validation 2003.

[101] A. Solar-Lezama, L. Tancau, R. Bodik, S. Seshia, and V. Saraswat, “Combinatorial sketching for finite programs,” in *Proceedings of the 12th International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS XII. New York, NY, USA: ACM, 2006, pp. 404–415.

[102] J. H. Perkins, S. Kim, S. Larsen, S. Amarasinghe, J. Bachrach, M. Carbin, C. Pacheco, F. Sherwood, S. Sidiroglou, G. Sullivan, W.-F. Wong, Y. Zibin, M. D. Ernst, and M. Rinard, “Automatically patching errors in deployed software,” in *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles (SOSP ’09)*. New York, NY, USA: ACM, 2009, pp. 87–102.

[103] B. Demsky, M. D. Ernst, P. J. Guo, S. McCamant, J. H. Perkins, and M. Rinard, “Inference and enforcement of data structure consistency specifications,” in *Proceedings of the 2006 International Symposium on Software Testing and Analysis*, ser. ISSTA ’06. New York, NY, USA: ACM, 2006, pp. 233–244.

[104] K. Chandy and J. Misra, “Distributed simulation: A case study in design and verification of distributed programs,” *Software Engineering, IEEE Transactions on*, vol. SE-5, no. 5, pp. 440–452, Sep. 1979.

[105] J. Misra and K. Chandy, “Proofs of networks of processes,” *Software Engineering, IEEE Transactions on*, vol. SE-7, no. 4, pp. 417–426, Jul. 1981.

[106] K. M. Chandy and L. Lamport, “Distributed snapshots: determining global states of distributed systems,” *ACM Trans. Comput. Syst.*, vol. 3, no. 1, pp. 63–75, 1985.

[107] S. Minopoli and G. Frehse, “Non-convex invariants and urgency conditions on linear hybrid automata,” in *Formal Modeling and Analysis of Timed Systems - 12th International Conference, FORMATS*, 2014, pp. 176–190.

[108] D. Seto, B. Krogh, L. Sha, and A. Chutinan, “The simplex architecture for safe on-line control system upgrades,” in *Proc. American Control Conference*, Philadelphia, PA, June 1998, pp. 3504–3508.

[109] ——, “Dynamic control system upgrade using the simplex architecture,” *Control Systems Magazine, IEEE*, vol. 18, no. 4, pp. 72–80, Aug. 1998.

[110] T. L. Crenshaw, E. Gunter, C. L. Robinson, L. Sha, and P. R. Kumar, “The simplex reference model: Limiting fault-propagation due to unreliable components in cyber-physical system architectures,” in *RTSS ’07*, Washington, DC, USA, 2007, pp. 400–412.

[111] X. Liu, Q. Wang, S. Gopalakrishnan, W. He, L. Sha, H. Ding, and K. Lee, "ORTEGA: An efficient and flexible online fault tolerance architecture for real-time control systems," vol. 4, no. 4, pp. 213–224, Nov. 2008.

[112] S. Bak, D. K. Chivukula, O. Adekunle, M. Sun, M. Caccamo, and L. Sha, "The system-level simplex architecture for improved real-time embedded system safety," in *Real-Time and Embedded Technology and Applications Symposium, IEEE*, vol. 0. Los Alamitos, CA, USA: IEEE Computer Society, 2009, pp. 99–107.

[113] S. Bak, A. Greer, and S. Mitra, "Hybrid cyberphysical system verification with simplex using discrete abstractions," in *Real-Time and Embedded Technology and Applications Symposium, IEEE*, vol. 0. Los Alamitos, CA, USA: IEEE Computer Society, 2010, pp. 143–152.

[114] P. S. Duggirala, T. T. Johnson, A. Zimmerman, and S. Mitra, "Static and dynamic analysis of timed distributed traces," in *Proceedings of the 33rd IEEE Real-Time Systems Symposium (RTSS 2012)*, San Juan, Puerto Rico, Dec. 2012.

[115] A. Zimmerman, "StarL for programming reliable robotic networks," Master's thesis, University of Illinois at Urbana-Champaign, 2012.

[116] H.-D. Tran, L. V. Nguyen, W. Xiang, and T. T. Johnson, "Order-reduction abstractions for safety verification of high-dimensional linear systems," *Discrete Event Dynamic Systems (DEDS)*, 2017. [Online]. Available: http://rdcu.be/q8Xd

[117] T. T. Johnson, S. Bak, M. Caccamo, and L. Sha, "Real-time reachability for verified simplex design," *ACM Transactions on Embedded Computing Systems (TECS)*, Feb. 2016.

[118] H.-D. Tran, L. V. Nguyen, and T. T. Johnson, "Large-scale linear systems from order-reduction (benchmark proposal)," in *3rd Applied Verification for Continuous and Hybrid Systems Workshop (ARCH)*, Vienna, Austria, Apr. 2016.

[119] O. A. Beg, L. V. Nguyen, A. Davoudi, and T. T. Johnson, "Computer-aided formal verification of power electronics circuits," in *8th International Workshop on Frontiers in Analog CAD (FAC)*, Frankfurt, Germany, 2017.

[120] O. A. Beg, A. Davoudi, and T. T. Johnson, "Reachability analysis of transformer-isolated dc-dc converters (benchmark proposal)," in *4th Applied Verification for Continuous and Hybrid Systems Workshop (ARCH)*, Pittsburgh, PA, Apr. 2017.

[121] A. Sogokon, K. Ghorbal, and T. T. Johnson, "Non-linear continuous systems for safety verification (benchmark proposal)," in *3rd Applied Verification for Continuous and Hybrid Systems Workshop (ARCH)*, Vienna, Austria, Apr. 2016.

[122] H.-D. Tran, L. V. Nguyen, W. Xiang, and T. T. Johnson, "Distributed autonomous systems (benchmark proposal)," in *4th Applied Verification for Continuous and Hybrid Systems Workshop (ARCH)*, Pittsburgh, PA, Apr. 2017.

[123] X. Leroy, "Formal verification of a realistic compiler," *Commun. ACM*, vol. 52, no. 7, pp. 107–115, Jul. 2009.

[124] G. Klein, K. Elphinstone, G. Heiser, J. Andronick, D. Cock, P. Derrin, D. Elkaduwe, K. Engelhardt, R. Kolanski, M. Norrish, T. Sewell, H. Tuch, and S. Winwood, "sel4: Formal verification of an os kernel," in *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles*, ser. SOSP '09. New York, NY, USA: ACM, 2009, pp. 207–220.

[125] G. Klein, "From a verified kernel towards verified systems," in *Programming Languages and Systems*, ser. Lecture Notes in Computer Science, K. Ueda, Ed. Springer Berlin Heidelberg, 2010, vol. 6461, pp. 21–33.

[126] T. A. L. Sewell, M. O. Myreen, and G. Klein, “Translation validation for a verified os kernel,” in *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation*, ser. PLDI ’13. New York, NY, USA: ACM, 2013, pp. 471–482.

[127] G. Klein, J. Andronick, K. Elphinstone, T. Murray, T. Sewell, R. Kolanski, and G. Heiser, “Comprehensive formal verification of an os microkernel,” *ACM Trans. Comput. Syst.*, vol. 32, no. 1, pp. 2:1–2:70, Feb. 2014.

[128] *ISO 26262-3: Road vehicles - Functional safety*, International Organization for Standardization (ISO) Std. ISO 26262, 2011.

[129] *Guidelines for the safety analysis of vehicle-based programmable systems (MISRA C:2012)*, Motor Industry Software Reliability Association (MISRA) Std., Mar. 2012. [Online]. Available: http://www.misra-c.com/

[130] J. W. Nimmer and M. D. Ernst, “Automatic generation of program specifications,” in *Proceedings of the 2002 ACM SIGSOFT international symposium on Software testing and analysis*, ser. ISSTA ’02. New York, NY, USA: ACM, 2002, pp. 229–239.

[131] M. Boshernitsan, R. Doong, and A. Savoia, “From Daikon to Agitator: Lessons and challenges in building a commercial tool for developer testing,” in *Proceedings of the 2006 international symposium on Software testing and analysis*, ser. ISSTA ’06. New York, NY, USA: ACM, 2006, pp. 169–180.

[132] C. Csallner, N. Tillmann, and Y. Smaragdakis, “DySy: Dynamic symbolic execution for invariant inference,” in *Software Engineering, 2008. ICSE ’08. ACM/IEEE 30th International Conference on*, 2008, pp. 281–290.

[133] R. M. Hierons, K. Bogdanov, J. P. Bowen, R. Cleaveland, J. Derrick, J. Dick, M. Gheorghe, M. Harman, K. Kapoor, P. Krause, G. Lüttgen, A. J. H. Simons, S. Vilkomir, M. R. Woodward, and H. Zedan, “Using formal specifications to support testing,” *ACM Comput. Surv.*, vol. 41, no. 2, pp. 9:1–9:76, Feb. 2009.

[134] Y. Annpureddy, C. Liu, G. Fainekos, and S. Sankaranarayanan, “S-taliro: A tool for temporal logic falsification for hybrid systems,” in *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2011.

[135] [Online]. Available: https://sites.google.com/a/asu.edu/s-taliro/s-taliro

[136] A. Donzé, “Breach, a toolbox for verification and parameter synthesis of hybrid systems,” in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, T. Touili, B. Cook, and P. Jackson, Eds. Springer Berlin / Heidelberg, 2010, vol. 6174, pp. 167–170.

[137] [Online]. Available: http://www.eecs.berkeley.edu/~donze/breach_page.html

[138] H. Yang, B. Hoxha, and G. Fainekos, “Querying parametric temporal logic properties on embedded systems,” in *International Conference on Testing Software and Systems*, ser. Lecture Notes in Computer Science, B. Nielsen and C. Weise, Eds. Springer Berlin Heidelberg, 2012, vol. 7641, pp. 136–151.

[139] P. S. Duggirala, S. Mitra, and M. Viswanathan, “Verification of annotated models from executions,” in *Proceedings of the Eleventh ACM International Conference on Embedded Software (EMSOFT ’13)*. Piscataway, NJ, USA: IEEE Press, 2013.

[140] F. Bernardini, M. Gheorghe, F. J. Romero-Campero, and N. Walkinshaw, “A hybrid approach to modeling biological systems,” in *Membrane Computing*, ser. LNCS, G. Eleftherakis, P. Kefalas, G. Paun, G. Rozenberg, and A. Salomaa, Eds. Springer Berlin Heidelberg, 2007, vol. 4860, pp. 138–159.

[141] T. N. Win, M. D. Ernst, S. J. Garland, D. Kyrly, and N. A. Lynch, “Using simulated execution in verifying distributed algorithms,” in *Verification, Model Checking, and Abstract*

Interpretation, ser. Lecture Notes in Computer Science, L. D. Zuck, P. C. Attie, A. Cortesi, and S. Mukhopadhyay, Eds. Springer Berlin Heidelberg, 2003, vol. 2575, pp. 283–297.

[142] N. Lynch, "Building blocks for high performance, fault-tolerant distributed systems," Massachusetts Institute of Technology, Tech. Rep. AFRL-SR-AR-TR-04-0138, Feb. 2004.

[143] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, pp. 183–235, 1994.

[144] S. Mitra, "A verification framework for hybrid systems," Ph.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA 02139, Sep. 2007.

[145] K.L. McMillan, *Symbolic Model Checking*. Norwell Massachusetts: Kluwer Academic Publishers, 1993.

[146] C. M. University, "Symbolic Model Verifier," www.cs.cmu.edu/ modelcheck/ smv.html, 2009.

[147] S. Owre, S. Rajan, J. Rushby, N. Shankar, and M. Srivas, "PVS: Combining specification, proof checking, and model checking," in *Computer-Aided Verification, CAV '96*, ser. LNCS, R. Alur and T. A. Henzinger, Eds., no. 1102. New Brunswick, NJ: Springer-Verlag, July/August 1996, pp. 411–414.

[148] G. J. Holzmann, *The SPIN Model Checker: Primer and Reference Manual*. Addison-Wesley Professional, Sep. 2003.

[149] G. Holzmann, "The model checker SPIN," *Software Engineering, IEEE Transactions on*, vol. 23, no. 5, pp. 279–295, May 1997.

[150] R. E. Bryant, "Symbolic Boolean manipulation with ordered binary-decision diagrams," *ACM Comput. Surv.*, vol. 24, no. 3, pp. 293–318, Sep. 1992.

[151] J. Moller, J. Lichtenberg, H. Andersen, and H. Hulgaard, "Difference decision diagrams," in *Computer Science Logic*, ser. Lecture Notes in Computer Science, J. Flum and M. Rodriguez-Artalejo, Eds. Springer Berlin / Heidelberg, 2009, vol. 1683, pp. 826–826.

[152] E. Goldberg, M. Prasad, and R. Brayton, "Using sat for combinational equivalence checking," in *Design, Automation and Test in Europe, 2001. Conference and Exhibition 2001. Proceedings*, 2001, pp. 114–121.

[153] A. Biere, A. Cimatti, E. Clarke, and Y. Zhu, "Symbolic model checking without BDDs," in *Tools and Algorithms for the Construction and Analysis of Systems*, ser. Lecture Notes in Computer Science, W. Cleaveland, Ed. Springer Berlin / Heidelberg, 1999, vol. 1579, pp. 193–207.

[154] H. Ganzinger, G. Hagen, R. Nieuwenhuis, A. Oliveras, and C. Tinelli, "DPLL(T): Fast decision procedures," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, R. Alur and D. Peled, Eds. Springer Berlin / Heidelberg, 2004, vol. 3114, pp. 293–295.

[155] R. Nieuwenhuis, A. Oliveras, and C. Tinelli, "Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T)," *J. ACM*, vol. 53, no. 6, pp. 937–977, Nov. 2006.

[156] C. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli, "Satisfiability modulo theories," *Handbook of Satisfiability*, pp. 737–797, 2008.

[157] L. Moura and N. Bjørner, "Satisfiability modulo theories: An appetizer," in *Formal Methods: Foundations and Applications*, ser. LNCS, M. M. Oliveira and J. Woodcock, Eds. Springer Berlin Heidelberg, 2009, vol. 5902, pp. 23–36.

[158] H. Mangassarian, A. Veneris, and M. Benedetti, "Robust QBF encodings for sequential circuits with applications to verification, debug, and test," *Computers, IEEE Transactions on*, vol. 59, no. 7, pp. 981–994, Jul. 2010.

[159] S. Yovine, “KRONOS: a verification tool for real-time systems,” *International Journal on Software Tools for Technology Transfer*, vol. 1, no. 1-2, pp. 123–133, 1997.

[160] R. Alur, R. Grosu, Y. Hur, V. Kumar, and I. Lee, “Modular specification of hybrid systems in Charon,” in *Hybrid Systems: Computation and Control*, ser. LNCS, N. Lynch and B. Krogh, Eds. Springer Berlin Heidelberg, 2000, vol. 1790, pp. 6–19.

[161] A. Chutinan and B. Krogh, “Computational techniques for hybrid system verification,” *Automatic Control, IEEE Transactions on*, vol. 48, no. 1, pp. 64–75, Jan. 2003.

[162] A. Kurzhanskiy and P. Varaiya, “Ellipsoidal toolbox,” in *45th IEEE Conference on Decision and Control (CDC)*, Dec. 2006, pp. 1498–1503.

[163] [Online]. Available: http://symbolaris.com/info/KeYmaera.html

[164] A. Platzer and E. Clarke, “Formal verification of curved flight collision avoidance maneuvers: A case study,” in *Formal Methods*, ser. LNCS, A. Cavalcanti and D. Dams, Eds. Springer, 2009, vol. 5850, pp. 547–562.

[165] C. L. Guernic and A. Girard, “Reachability analysis of linear systems using support functions,” *Nonlinear Analysis: Hybrid Systems*, vol. 4, no. 2, pp. 250–262, 2010.

[166] D. Dill, “What’s between simulation and formal verification?” in *Design Automation Conference, 1998. Proceedings*, June 1998, pp. 328–329.

[167] N. Beckman, A. Nori, S. Rajamani, R. Simmons, S. Tetali, and A. Thakur, “Proofs from tests,” *Software Engineering, IEEE Transactions on*, vol. 36, no. 4, pp. 495–508, 2010.

[168] K. Lata and S. K. Roy, “Formal verification of analog and mixed signal designs using SPICE circuit simulation traces,” *Journal of Electronic Testing*, pp. 1–26, 2013.

[169] S. Gao, J. Avigad, and E. Clarke, “Delta-decidability over the reals,” in *Logic in Computer Science (LICS), 2012 27th Annual IEEE Symposium on*, 2012, pp. 305–314.

[170] A. Eggers, M. Fränzle, and C. Herde, “SAT modulo ODE: A direct SAT approach to hybrid systems,” in *Automated Technology for Verification and Analysis*, ser. Lecture Notes in Computer Science, S. Cha, J.-Y. Choi, M. Kim, I. Lee, and M. Viswanathan, Eds. Springer Berlin / Heidelberg, 2008, vol. 5311, pp. 171–185.

[171] A. Cimatti, S. Mover, and S. Tonetta, “A quantifier-free smt encoding of non-linear hybrid automata,” in *Formal Methods in Computer-Aided Design (FMCAD), 2012*, 2012, pp. 187–195.

[172] M. Pajic, Z. Jiang, I. Lee, O. Sokolsky, and R. Mangharam, “From verification to implementation: A model translation tool and a pacemaker case study,” in *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2012 IEEE 18th*. IEEE, 2012, pp. 173–184.

[173] Z. Jiang, M. Pajic, R. Alur, and R. Mangharam, “Closed-loop verification of medical devices with model abstraction and refinement,” *International Journal on Software Tools for Technology Transfer*, vol. 16, no. 2, pp. 191–213, 2014.

[174] M. Pajic, R. Mangharam, O. Sokolsky, D. Arney, J. Goldman, and I. Lee, “Model-driven safety analysis of closed-loop medical systems,” *Industrial Informatics, IEEE Transactions on*, vol. 10, no. 1, pp. 3–16, 2014.

[175] M. Pajic, Z. Jiang, I. Lee, O. Sokolsky, and R. Mangharam, “Safety-critical medical device development using the upp2sf model translation tool,” *ACM Trans. Embed. Comput. Syst.*, vol. 13, no. 4s, pp. 127:1–127:26, Apr. 2014. [Online]. Available: http://doi.acm.org/10.1145/2584651

[176] A. Agrawal, G. Simon, and G. Karsai, “Semantic translation of simulink/stateflow models to hybrid automata using graph transformations,” *Electronic Notes in Theoretical Computer Science*, vol. 109, pp. 43–56, 2004.

[177] L. Carloni, M. D. Di Benedetto, A. Pinto, and A. Sangiovanni-Vincentelli, “Modeling techniques, programming languages, design toolsets and interchange formats for hybrid systems,” Tech. Rep., 2004.

[178] A. Pinto, A. L. Sangiovanni-Vincentelli, L. P. Carloni, and R. Passerone, “Interchange formats for hybrid systems: Review and proposal,” in *Hybrid Systems: Computation and Control*, ser. LNCS, M. Morari and L. Thiele, Eds. Springer Berlin Heidelberg, 2005, vol. 3414, pp. 526–541.

[179] B. Meenakshi, A. Bhatnagar, and S. Roy, “Tool for translating simulink models into input language of a model checker,” in *Formal Methods and Software Engineering*, ser. Lecture Notes in Computer Science, Z. Liu and J. He, Eds. Springer Berlin Heidelberg, 2006, vol. 4260, pp. 606–620. [Online]. Available: http://dx.doi.org/10.1007/11901433_33

[180] A. Pinto, L. Carloni, R. Passerone, and A. Sangiovanni-Vincentelli, “Interchange format for hybrid systems: Abstract semantics,” in *Hybrid Systems: Computation and Control*, ser. LNCS, J. P. Hespanha and A. Tiwari, Eds. Springer Berlin Heidelberg, 2006, vol. 3927, pp. 491–506.

[181] L. P. Carloni, R. Passerone, A. Pinto, and A. L. Sangiovanni-Vincentelli, “Languages and tools for hybrid systems design,” *Foundations and Trends in Electronic Design Automation*, vol. 1, 2006.

[182] D. van Beek, M. Reniers, R. Schiffelers, and J. Rooda, “Foundations of a compositional interchange format for hybrid systems,” in *Hybrid Systems: Computation and Control*, ser. LNCS, A. Bemporad, A. Bicchi, and G. Buttazzo, Eds. Springer Berlin Heidelberg, 2007, vol. 4416, pp. 587–600.

[183] M. Li and R. Kumar, “Model-based automatic test generation for simulink/stateflow using extended finite automaton,” in *Automation Science and Engineering (CASE), 2012 IEEE International Conference on*, Aug. 2012, pp. 857–862.

[184] P. Schrammel and B. Jeannet, “From hybrid data-flow languages to hybrid automata: A complete translation,” in *Proceedings of the 15th ACM International Conference on Hybrid Systems: Computation and Control*, ser. HSCC ’12. New York, NY, USA: ACM, 2012, pp. 167–176.

[185] M. Li and R. Kumar, “Reduction of automated test generation for simulink/stateflow to reachability and its novel resolution,” in *Automation Science and Engineering (CASE), 2013 IEEE International Conference on*, Aug. 2013, pp. 1089–1094.

[186] D. N. Agut, D. van Beek, and J. Rooda, “Syntax and semantics of the compositional interchange format for hybrid systems,” *The Journal of Logic and Algebraic Programming*, vol. 82, no. 1, pp. 1 – 52, 2013.

[187] S. Ran, J. Lin, Y. Wu, J. Zhang, and Y. Xu, “Converting ptolemy ii models to spaceex for applied verification,” in *Algorithms and Architectures for Parallel Processing*, ser. LNCS, X.-h. Sun, W. Qu, I. Stojmenovic, W. Zhou, Z. Li, H. Guo, G. Min, T. Yang, Y. Wu, and L. Liu, Eds. Springer International Publishing, 2014, vol. 8630, pp. 669–683.

[188] G. Hamon and J. Rushby, “An operational semantics for stateflow,” *International Journal on Software Tools for Technology Transfer*, vol. 9, no. 5-6, pp. 447–456, 2007.

[189] C. Chen, J. Sun, Y. Liu, J. Dong, and M. Zheng, “Formal modeling and validation of stateflow diagrams,” *International Journal on Software Tools for Technology Transfer*, vol. 14, no. 6, pp. 653–671, 2012.

[190] E. T. Inc, “Scade suite.” [Online]. Available: http://www.esterel-technologies.com/-products/scade-suite/

[191] C. Zhou and R. Kumar, "Semantic translation of simulink diagrams to input/output extended finite automata," *Discrete Event Dynamic Systems*, vol. 22, no. 2, pp. 223–247, 2012.

[192] K. Manamcheri, S. Mitra, S. Bak, and M. Caccamo, "A step towards verification and synthesis from Simulink/Stateflow models," in *Proc. of the 14th Intl. Conf. on Hybrid Systems: Computation and Control (HSCC)*. ACM, 2011, pp. 317–318.

[193] [Online]. Available: https://wiki.cites.illinois.edu/wiki/display/MitraResearch/HyLink

[194] K. Manamcheri Sukumar, "Translation of simulink-stateflow models to hybrid automata," Master's thesis, University of Illinois at Urbana-Champaign, 2011.

[195] L. V. Nguyen, C. Schilling, S. Bogomolov, and T. T. Johnson, "Runtime verification of model-based development environments," in *15th International Conference on Runtime Verification (<a href="http://rv2015.conf.tuwien.ac.at/">RV 2015</a>)*, Vienna, Austria, Sep. 2015.

[196] M. U. Sardar, N. Afaq, K. A. Hoque, T. T. Johnson, and O. Hasan, "Probabilistic formal verification of the sats concept of operation," in *Proceedings of the 8th NASA Formal Methods (<a href=*, S. Rayadurgam and O. Tkachuk, Eds.

[197] S. A. Chowdhury, T. T. Johnson, and C. Csallner, "Cyfuzz: : A differential testing framework for cyber-physical systems development environments," in *6th International Workshop Workshop on Design, Modeling and Evaluation of Cyber Physical Systems (<a href="http://www.cyphy.org/">CyPhy 2016</a>)*, Pittsburgh, PA, Aug. 2016.

[198] A. Sogokon, K. Ghorbhal, and T. T. Johnson, "Operational models of piecewise-smooth systems," in *17th ACM SIGBED International Conference on Embedded Software (EMSOFT 2017)*, Oct. 2017.

[199] S. Bak and T. T. Johnson, "Periodically-scheduled controller analysis using hybrid systems reachability and continuization," in *36th IEEE Real-Time Systems Symposium (RTSS)*. San Antonio, Texas: IEEE Computer Society, Dec. 2015.

[200] S. Bak, S. Bogomolov, T. A. Henzinger, T. T. Johnson, and P. Prakash, "Scalable static hybridization methods for analysis of nonlinear systems," in *Proc. of the 19th Intl. Conf. on Hybrid Systems: Computation and Control (HSCC)*. ACM, Apr. 2016.

[201] O. A. Beg, A. Davoudi, and T. T. Johnson, "Charge pump phase-locked loops and full wave rectifiers for reachability analysis (benchmark proposal)," in *3rd Applied Verification for Continuous and Hybrid Systems Workshop (ARCH)*, Vienna, Austria, Apr. 2016.

[202] L. V. Nguyen, D. Maksimovic, T. T. Johnson, and A. Veneris, "Quantified bounded model checking for rectangular hybrid automata," in *9th International Workshop on Constraints in Formal Verification (CFV 2015)*, Austin, Texas, Nov. 2015.

[203] A. Sogokon, P. Jackson, and T. T. Johnson, "Verifying safety and persistence properties of hybrid systems using flowpipes and continuous invariants," in *9th NASA Formal Methods Symposium (NFM 2017)*, May 2017.

[204] A. Sogokon, K. Ghorbal, and T. T. Johnson, "Decoupled simulating abstractions of nonlinear ordinary differential equations," in *Proceedings of the 21st International Symposium on Formal Methods (<a href="http://fm2016.cs.ucy.ac.cy">FM 2016</a>)*, Limassol, Cyprus, Dec. 2016.

[205] W. Xiang, H.-D. Tran, and T. T. Johnson, "Reachable set estimation and control for switched linear systems with dwell-time restriction," in *Proceedings of the 55th IEEE Conference on Decision and Control (<a href="http://cdc2016.ieeecss.org/">CDC 2016</a>)*, Las Vegas, NV, USA, Dec. 2016.

[206] W. Xiang, “EnglishEvent-triggered control for continuous-time switched linear systems,” *EnglishIET Control Theory and Applications*, February 2017.

[207] W. Xiang and T. T. Johnson, “On reachable set estimation for discrete-time switched linear systems under arbitrary switching,” in *30th IEEE American Control Conference (ACC 2017)*, 2017.

[208] W. Xiang, H.-D. Tran, and T. T. Johnson, “Output reachable set estimation for switched linear systems and its application in safety verification,” *IEEE Transactions on Automatic Control (TAC)*, 2017.

[209] O. A. Beg, H. Abbas, T. T. Johnson, and A. Davoudi, “EnglishModel validation of pwm dc-dc converters,” *EnglishIEEE Transactions on Industrial Electronics. (TIE)*, June 2017.

7. Appendices

All publications resulting from this project are enclosed =[35-37, 54, 117, 119-121, 195-203] [51, 116, 118, 122, 204-209].

Periodically-Scheduled Controller Analysis using Hybrid Systems Reachability and Continuization

Stanley Bak
Air Force Research Lab - Information Directorate
Rome, NY, USA

Taylor T. Johnson
University of Texas at Arlington
Arlington, TX, USA

***Abstract*—Cyber-physical systems (CPS) consist of physical entities that obey dynamical laws and interact with software components. A typical CPS implementation includes a discrete controller, where software periodically samples physical state and produces actuation commands according to a real-time schedule. Such a hybrid system can be modeled formally as a hybrid automaton. However, reachability tools to verify specifications for hybrid automata do not perform well on such periodically-scheduled models. This is due to a combination of the large number of discrete jumps and the nondeterminism of the exact controller start time. In this paper, we demonstrate this problem and propose a solution, which is a validated abstraction mechanism where every behavior of the original sampled system is contained in the behaviors of a purely continuous system with an additive nondeterministic input. Reachability tools for hybrid automata can better handle such systems. We further improve the analysis by considering local analysis domains. We automate the proposed technique in the Hyst model transformation tool, and demonstrate its effectiveness in a case study analyzing the design of a yaw-damper for a jet aircraft.**

I. Introduction

Periodic real-time scheduling is a widespread method used to control a physical plant as part of a cyber-physical system (CPS). Typical schedulers, such as rate-monotonic (RM) or earliest deadline first (EDF) [1], give a guarantee of *periodic* execution. In each period, sensors are read, the control algorithm is run, and actuator outputs are set. The physical world, on the other hand, evolves continuously. Models of the physical world may be given using differential equations that are obeyed at all times.

In this work, we analyze the periodically-scheduled controller subsystems of CPS using hybrid automata [2] and associated analysis tools. A hybrid automaton can directly model both the continuous behaviors and discrete aspects that arise when real-time scheduling and sampled control is combined with a continuously-evolving physical plant. The set of reachable states of a hybrid automaton, if it can be computed or overapproximated, can be used to formally prove control-theoretic properties about the system's transient and steady-state behavior. The controller subsystem models, after being proven correct, could then be integrated with hybrid automaton models of other parts of the system using modeling methods like hybrid input/output automata (HIOA) [3]. Reasoning about properties of the combined system could then be performed using assume-guarantee reasoning [4]. With such hybrid systems analysis, properties can be formally proven about *sets* of initial states as well as behaviors under bounded sensor error, actuator error, and other uncertainties. This has the potential to detect errors not found during simulation and testing, which deal with single initial states and specific execution traces.

Directly analyzing the controller subsystems of CPS using hybrid automaton reachability tools, unfortunately, does not usually work. One issue is that a large number of controller updates need to be considered in the analysis. The control code may need to be run tens or hundreds of times a second, and the physical system may need to evolve for tens of seconds to show the properties of interest. The number of discrete transitions that occur thus becomes extremely large. Real-time schedulers may also have variability in the exact scheduling time of the controller. Hybrid automaton reachability analysis tools perform poorly in such cases, with error bounds growing unacceptably large in the presence of many discrete transitions and timing uncertainty [5], [6].

In order to overcome these challenges, we apply a variant of the continuization technique [7], where a fast-switching hybrid system is abstracted by a continuous system with an additive nondeterministic input. We provide theoretical methods to compute bounds on the nondeterminism input needed for the continuization of periodically-scheduled controllers, which is essential for abstraction soundness. The developed approach is then automated using the Hyst [8] model transformation tool. In this way, we provide both a theoretical method that enables controller analysis with hybrid automaton reachability tools, and a practical way to use it.

The main contributions of this paper are:

- the modeling of periodically-controlled CPS using hybrid automata, with several models proposed based on possible implementation variations,
- the validated use of continuization to enable the analysis of these models, and a theoretical method to compute the bound on the nondeterminism globally as well as within local analysis domains,
- the implementation of the proposed technique in the Hyst model transformation tool, which allows rapid application to new hybrid automaton models, and
- a demonstration of the effectiveness of the proposed analysis approach on the design of a yaw damper system for a 747 jet aircraft.

DISTRIBUTION A. Approved for public release; Distribution unlimited. (Approval AFRL PA #88ABW-2015-2401, 15 MAY 2015)

In the next section, we present a brief background on modeling hybrid systems, give direct approaches for modeling real-time-scheduled controllers with hybrid automata, and provide reachability results showing scalability issues with these direct models. Next, Sec. III describes continuization and methods for computing the nondeterministic bounds it uses, which are essential for method accuracy. Then Sec. IV briefly describes the Hyst model transformation tool and the illustrates the continuization pass that implements the technique developed in this paper. Sec. V provides a case study showing the advantage of the analysis on a yaw damper control system for a 747 aircraft. A brief discussion of related techniques, especially a comparison versus classical control theoretic methods is given in Sec. VI, followed by a conclusion.

II. Hybrid Systems Modeling

The controller subsystem of a cyber-physical system (CPS) consists of a physical system interacting with a software controller, running periodically on a system using a real-time scheduler. A specific implementation can be formalized using a hybrid automaton model, and then its behavior, as well as the behavior of a composition of these subsystem models, can be analyzed using hybrid automata reachability tools. In this section, we elaborate on modeling controller subsystems using the hybrid automaton formalism. We first review hybrid automata (Sec. II-A), then propose three models that capture different possible implementations of a controller subsystem of a CPS (Sec. II-B). Finally, we attempt to directly perform reachability analysis of these systems using reachability analysis tools (Sec. II-C), which is shown to be challenging.

A. Preliminaries

A hybrid automaton is a formal model that captures both discrete behaviors as well as continuous dynamics present in a hybrid system. Roughly, it is a finite state machine with ordinary differential equations defined in each mode for a set of real-valued continuous variables.

Definition 1 (Hybrid Automaton)**.** *A* Hybrid Automaton *is a tuple*
$\mathcal{H}$ = (Loc, Var, Init, Flow, Trans, Inv) that defines:

- *a finite set of locations Loc,*
- *a set of n real-valued continuous variables Var = $\{x_1, \ldots, x_n\}$,*
- *an initial condition Init $\subseteq \mathbb{R}^n$ for each $\ell \in$ Loc,*
- *for each location ℓ, a relation Flow(ℓ) relating variables and their derivatives,*
- *a set of discrete transitions Trans, where each element is a tuple (ℓ, g, r, ℓ') with source location ℓ, guard g given as constraint on $\mathbb{R}^n$, reset r given as a function from $\mathbb{R}^n$to $\mathbb{R}^n$, and destination location ℓ',*
- *an invariant Inv(ℓ) $\subseteq \mathbb{R}^n$ for each location ℓ.*

A state of a hybrid system is a tuple $(\ell, \mathbf{X})$, where the discrete state is $\ell \in$ *Loc* and the continuous state $\mathbf{X}$ is a valuation—a mapping from a variable name to a point in the reals—of the continuous variables in *Var*.

Definition 2 (Trajectory)**.** *A* trajectory *of a hybrid system is an alternating sequence of continuous evolutions and discrete transitions, starting from a state in Init. Trajectories are subject to the following restrictions:*

- *the first state of the trajectory is an element of Init,*
- *during each continuous evolution, the continuous state evolves over an interval of real-valued time in accordance with the differential equations defined by Flow,*
- *during each continuous evolution, the continuous states always satisfy the location's invariant*[1]*, and*
- *during each discrete transition, the prestate is contained in transition's guard, and the change in state corresponds to applying the reset function to the continuous prestate and updating the location to ℓ'.*

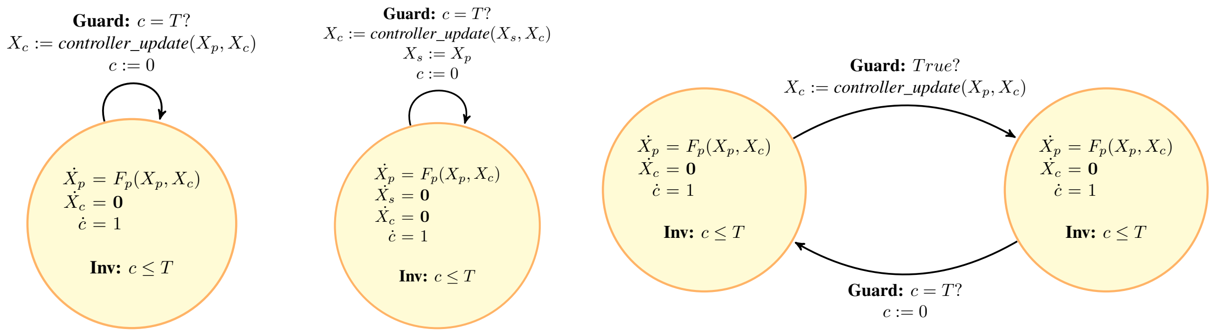
Definition 3 (Reachable Set)**.** *The set of all states that exist in any trajectory is called the* reachable set*. For a given hybrid automaton $\mathcal{H}$, we use $\textsc{reach}(\mathcal{H})$ to denote the reachable set of $\mathcal{H}$. Given a subset of the variables $Y \subseteq$* Var *of hybrid automaton $\mathcal{H}$, the reach set projected onto those variables is written as $\textsc{reach}(\mathcal{H}) \downarrow Y$. Typically we will be concerned with time-bounded reachable sets, where the amount of time that has elapsed during the continuous evolution portions of each trajectory is less than or equal to some given bound.*

B. CPS Modeling

We now describe three different ways that a CPS controller subsystem can be modeled using the hybrid automaton formalism, which correspond to different possible system implementations. First, we introduce the notion of a *Sampled CPS*, which has a continuous portion governed by differential equations, and a *controller_update* function that updates the discretely-controlled variables.

Definition 4 (Sampled CPS)**.** *A* Sampled CPS *is a system with n continuous variables divided into two groups. The first $np \leq n$ variables are the physical variables, and the remaining $nc = n - np$ variables are the cyber variables. The set of variables $Var = \{x_1, x_2, \ldots, x_n\}$ is partitioned into physical variables $X_p = \{p_1, p_2, \ldots, p_{np}\}$ and cyber variables $X_c = \{c_1, c_2, \ldots, c_{nc}\}$, where each variable $x_i \in \mathbb{R}$. Each physical variable has an associated differential equation, $\dot{p}_1 = f_1$, $\dot{p}_2 = f_2$, ..., $\dot{p}_{np} = f_{np}$, where each $\dot{p}_i = f_i$ is a function $\mathbb{R}^n \rightarrow \mathbb{R}$. To ensure existence and uniqueness of the solutions, the differential equations are assumed to be Lipschitz continuous in the domain of interest. The dynamics for the physical variables are provided, so $F_p = (f_1, f_2, \ldots, f_{np})$ is given. The remaining nc variables are set periodically in control software, and remain constant between updates (zero-order hold). Their differential equations are given as $\dot{c}_1 = 0$, $\dot{c}_2 = 0$, ..., $\dot{c}_{cn} = 0$. The control software is defined by a function controller_update : $\mathbb{R}^n \rightarrow \mathbb{R}^{nc}$, which updates the cyber variables based on the*

[1] If at some point the invariant were to become false, a discrete transition must be taken immediately. If no transition's guards are enabled, the model is said to deadlock as time cannot advance.



(a) Model 1 (b) Model 2 (c) Model 3

Fig. 1: Various hybrid automaton models formalize different implementations of a periodically-sampled CPS.

system state. The controller_update function can be decomposed into nc functions where each one updates a single cyber variable, $c_1 :=$ controller_update$_1(X_p, X_c), \ldots, c_{nc} :=$ controller_update$_{nc}(X_p, X_c)$. In this work, we will restrict the controller_update functions to ones that are differentiable and locally Lipschitz continuous in the input arguments, in the domain of interest (for example, discrete approximations of continuous controllers).

Model 1: The simplest model is for a strict periodic controller, where the control software runs with a given period, T. This could correspond to a system using a time-division multiple-access (TDMA) or other time-triggered scheduler, where the control task is nonpreemptive and the worst-case execution time (WCET) fairly short. In the model, a single location exists where time can elapse. An extra clock variable, c, is added to the hybrid automaton that ticks at rate one ($\dot{c} = 1$). When the clock reaches the period, a transition is forced by an invariant in the single location that $c \leq T$, which prevents continuous evolutions from continuing. The transition executes the controller logic when the clock reaches the period, then resets the clock to 0, and subsequently repeats periodically. A hybrid automaton visualization of this model is shown in Fig. 1(a). The strict periodic controller, however, does not exactly capture the behavior of a system using a real-time scheduler. A scheduler like rate-monotonic (RM) or earliest deadline first (EDF), provides a guarantee of execution *at some point within the period.*

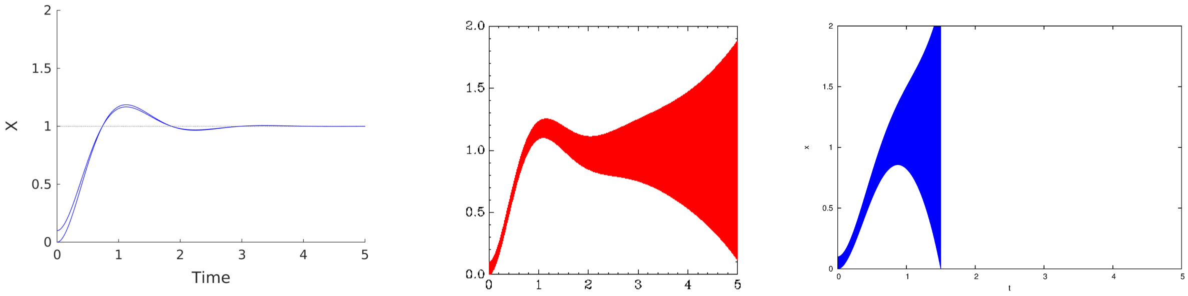
Model 2: An alternative implementation, which uses a real-time scheduler such as RM or EDF would sample the system at the start of the period, and write the actuation values at the end of the period. This can be modeled using a hybrid automaton by starting with the strictly periodic system (Model 1) and adding np additional cyber variables, which we call X_s, with derivatives equal to zero that model the sampled state. On the actuator assignment (*controller_update*) at the end of each period, the controller logic will then compute on the state sampled from the start of the period. After updating the cyber variables, the physical state would then be sampled again and stored into X_s for use at the end of the next period. The hybrid automaton model of this system is given in Fig. 1(b). The downside of such a controller is there is a one period delay introduced into the system, which may affect control performance, as well as np additional variables in the model, which may affect analysis scalability.

Model 3: An alternative implementation may consider directly sampling and actuating at some point during each period, where the sampling point is nondeterministic. This would be a reasonable model if the control task's execution is short and the task is non-preemptive. This model is similar to the strictly periodic Model 1 (Fig. 1(a)), except that: (1) a second mode is added to indicate if the controller has run yet during the period, (2) the first transition (the call to *controller_update*) happens nondeterministically up to the period T owing to the invariant $c \leq T$, and (3) the second transition (the end of the control period) happens when the clock reaches T time. The modified automaton is shown in Fig. 1(c). This model uses nondeterminism in discrete transitions to capture the type of guarantee provided by a real-time scheduler: that the control logic will execute and finish *at some point* within each period.

More complicated models could also be considered. For example, if the execution time was non-negligible or the task was preemptive, the state could be sampled nondeterministically at some point during the period minus the WCET, and then actuation could be performed nondeterministically up to the end of the period.

C. Preliminary Reachability Analysis

Although hybrid automata can model real-time scheduled controllers and plants as shown above, an important factor is tractability of analysis. Since analysis of even moderately-complicated hybrid automata is undecidable [9], tools often compute an overapproximation of the reachable states, which is sufficient for safety analysis (making sure unsafe states are not reachable). If the set of reachable states may be computed for unbounded time (if the reachability algorithm reaches a fixed-point) and the resulting set of states is bounded, then conclusions can also be drawn about system stability. In the presence of a large number of discrete switches, reachability analysis tools may significantly overapproximate the reachable set of states, due to the need to perform intersections of reachable sets with surfaces representing guard conditions [6]. These intersections are typically done geometrically, and result



(a) Simulations from $x \in \{0, 0.1\}$

(b) SpaceEx Reachability

(c) Flow* Reachability

Fig. 2: The response for the periodically-controlled double-integrator system from Example 1 converges in simulation, but appears to diverge during reachability analysis.

in an overapproximation of the actual intersection, introducing some error at every discrete transition. Due to this concern, we empirically evaluate the performance of two modern reachability tools, SpaceEx [5] and Flow* [10], [11], on a simple control system using the approach from Model 1 (Fig. 1(a)).

Example 1 (Double-Integrator System)**.** *A double-integrator system, such as point moving along a 1-d line controlled through its acceleration, has two physical variables: x, its position, v, its velocity, and a single cyber variables a, its acceleration. The dynamics are $\dot{x} = v$, $\dot{v} = a$, and the acceleration a is set periodically by the control logic. There is a fixed setpoint the system tries to move towards at $x = 1$. The acceleration is set using a PD controller with gains $P = 10$ and $D = 3$. The controller_update function periodically assigns $a := P * (1 - x) + D * -v$. The period of the control task is $T = 0.005$ seconds (200 Hz). The initial states are $x \in [0, 0.1]$ and $v = 0$.*

Using the system in Example 1, we construct the corresponding hybrid automaton (shown in the Appendix in Fig. 8) and examine the controller's response. A control Lyapunov function may be derived to show stabilization of the purely continuous system to the setpoint of $x = 1$ and $v = 0$. In Matlab simulations of the periodically-sampled system from the boundary of the initial states (from both $x = 0$ and $x = 0.1$), the system easily converges to the setpoint. When performing reachability, however, both SpaceEx and Flow* produce divergent reachable sets, due to overapproximation error introduced at each of the discrete transitions. The simulations and reachability visualization are shown in Fig. 2.

Although effort was taken to optimize various tool parameters, they could likely be further adjusted to get a slightly better response. For this particular system, if the tools had built-in support for time-triggered transitions and could infer that the clock acts as a time-trigger for the discrete transition, the error in the computation could likely be reduced (although we could not find time-triggered support in either tool's documentation). However, this would not work for the nondeterministic switch in Model 3 (Fig. 1(c)), since that discrete transition (invocation of *controller_update*) can occur at any time within the period, based on the guarantees provided by schedulers like RM and EDF. The problem of accumulated error in reachability from many discrete transitions, in general, cannot be eliminated.

III. Continuization for Improved Analysis

The occurrence of many discrete transitions leads to accumulated error during reachability analysis because of a need to repeatedly take intersections of sets of states with the transition guards. One idea to get better accuracy, therefore, is to eliminate the discrete transitions altogether. Intuitively, this process relies on the observation that the behavior of the periodically-sampled system is contained in the behavior of the continuously-controlled system with some additional bounded nondeterministic input.

This process of validated abstraction of the sampled hybrid automaton by a continuous one is called *continuization* [12], and is briefly reviewed in the next subsection (Sec. III-A). Here, we apply the continuization idea in order to analyze periodic control systems, which has not been done before. This process relies on having a bound on the speed of changes of the cyber variables, and computing this bound is then described (Sec. III-B).

A. *Continuization*

Continuization is the process of abstracting a system with many discrete switches by a continuous one with an extra nondeterministic input. Previously, it was used to analyze rapidly-switching electric circuits [12], specifically locking time and stability properties for charge-pump phase-locked loops. The key challenge when performing continuization is determining the amount of nondeterministic input that is necessary in order to guarantee that all behaviors of the sampled system are captured by the continuous one, but not too much that analysis accuracy suffers.

In the earlier circuit work, this was done by solving for the change of state in one cycle with a known switching time. Since there was no closed-form solution for the switching time, interval analysis was performed using the ranges of possible switching times, and then this was used to derive conservative bounds on the change in state.

We want to apply continuization in order to analyze periodically-controlled CPS. We formalize this process by using sampling deviation functions.

Definition 5 (Sampling Deviation)**.** *A* sampling deviation ω_i *is a function* $\mathbb{R} \to \mathbb{R} \times \mathbb{R}$*, which, given a time, produces an upper and lower bound on the difference of a cyber variable* $c_i \in X_c$*, between its value in a sampled CPS and the update function controller_update*(X_p, X_c)*.*

Given a sampling deviation function ω_i for each cyber variable, we can construct an overapproximation of the sampled CPS. First, we construct a continuous approximation of the sampled CPS.

Definition 6 (Continuous Approximation)**.** *A* continuous approximation *of a sampled CPS is a hybrid automaton where the controller logic is run continuously. That is, the discrete update for each cyber variables in* $c_i \in X_c$ *is removed from the system, and each cyber variable's differential equation is set to* $\dot{c}_i = \frac{d}{dt}\textit{controller\_update}_i(X_p, X_c)$*. The variable* c_i*'s initial value is set to the value when the controller is run at the original initial state, controller_update*$_i(X_p(0), X_c(0))$*.*

The continuous approximation differs from the original sampled CPS. A *continuized abstraction* accounts for this difference by adding nondeterminism to every occurrence of each cyber variable within the continuous approximation.

Definition 7 (Continuized Abstraction, Continuization)**.** *A* continuized abstraction $\mathcal{H}_c$ *of a sampled CPS* $\mathcal{H}$ *is constructed starting from* $\mathcal{H}$*'s continuous approximation. Each occurrence of a cyber-variable* c_i *in the continuous approximation gets an extra term added equal to the sampling deviation* ω_i*. If any of the* ω_i *change over time, an additional time variable* t *is added to the system that starts at 0 and ticks at rate 1 forever.*

The model constructed using the above continuization approach will have trajectories of the physical variables that contain all the behaviors in the original sampled CPS.

Theorem 1 (Soundness of Continuization)**.** *Given a sampled CPS* $\mathcal{H}$ *as well as its continuized abstraction* $\mathcal{H}_c$*,* $\text{REACH}(\mathcal{H}) \downarrow X_p \subseteq \text{REACH}(\mathcal{H}_c) \downarrow X_p$*.*

Proof. Consider any cyber variable $c_i \in X_c$. Let $\text{Val}_{\text{sampled}}(c_i)$ be the value of the variable in the sampled CPS, and $\text{Val}_{\text{abstract}}(c_i)$ be the value of the variable in the continuized abstraction. At any time t in a trajectory, we first show that $\text{Val}_{\text{sampled}}(c_i) \in \text{Val}_{\text{abstract}}(c_i) + \omega_i(t)$.

By the definition of the sampling deviation function, the difference between $\text{Val}_{\text{sampled}}(c_i)$ and *controller_update*$_i(X_p, X_c)$ at time t must be contained in the interval $\omega_i(t)$. Therefore, $\text{Val}_{\text{sampled}}(c_i)$ is contained in *controller_update*$_i(X_p, X_c) + \omega_i(t)$. The continuous approximation at time t is equal to *controller_update*$_i(X_p, X_c)$, and by the construction of the continuized abstraction from the continuous approximation, the inclusion $\text{Val}_{\text{sampled}}(c_i) \in \text{Val}_{\text{abstract}}(c_i) + \omega_i(t)$ holds.

In the construction of the continuous abstraction, each cyber variable c_i in the continuous approximation was replaced by $c_i + \omega_i(t)$. Since, as shown above, $\text{Val}_{\text{sampled}}(c_i) \in \text{Val}_{\text{abstract}}(c_i) + \omega_i(t)$, the derivatives for every variable in the sampled CPS will be contained in the derivatives of continuized abstraction. In particular, the physical variable values in the continuized abstraction also contain the sampled CPS physical variable values. The discrete transitions between the two systems are identical, except for the removal of the periodic cyber-variable updates in the continuized abstraction. Thus, any discrete transition (other than controller updates, which only update cyber variables and for which we already showed containment) taken by the sampled CPS can also be taken by the continuized version. Since a trajectory is an alternating sequence of continuous evolutions and discrete transitions, and the initial states are the same, by induction on the length of a trajectory, the values of the physical variables in the sampled CPS are always contained in the values of the physical variables in the continuized abstraction. Therefore, $\text{REACH}(\mathcal{H}) \downarrow X_p \subseteq \text{REACH}(\mathcal{H}_c) \downarrow X_p$. □

B. Producing Sampling Deviation Functions

The key to continuization is to construct sampling deviation functions that provide an upper and lower bound on the difference of each cyber variable between the sampled CPS and the *controller_update* function. One way to compute such a function is by looking at the maximum rate of change (bounded by a Lipschitz constant) of the derivative of each cyber variable in the continuous approximation. This process makes use of standard interval arithmetic multiplication, $[a, b] * [c, d] = [\min(a*c, a*d, b*c, b*d), \max(a*c, a*d, b*c, b*d)]$.

Lemma 1 (Sampling Deviation using Lipschitz Constant)**.** *Given interval bounds,* $K = [K^{min}, K^{max}]$*, on the rate of change of the derivative of* c_i *in the continuous approximation, and the period of the associated strictly-periodic (Model 1 from Fig. 1) controller,* T*, a sampling deviation function is* $\omega_i = [-T, 0] * K$*.*

Proof. The sampling deviation function needs to bound the difference of the value of the variable c_i in a sampled CPS and *controller_update*$_i(X)$. The difference between *controller_update*$_i$ at the last sample time (which is the current value of the cyber variable in the sampled CPS), and *controller_update*$_i$ at the current time (which is its value in the continuous approximation) is at most a product of the maximum rate of change K, and the time since the last sample. The difference between the last controller update and the current time must be in the interval $[-T, 0]$, since it is a strictly periodic controller with period T. Assuming the first sample occurs at time 0, by induction on the number of samples, this property will hold for every sampling period and therefore over all time. □

In this case, we had considered a strictly periodic controller, such as the one given by Model 1 in Sec. II-B. To compute the function for a nondeterministic controller such as Model 3, all that would need to be adjusted is the time of the last controller update. In the worst case, a sample will occur at the start of one period, and at the end of the next period. In that case, the maximum time between updates is $2 * T$, so using an interval of $[-2 * T, 0]$ in Lemma 1 would be adequate.

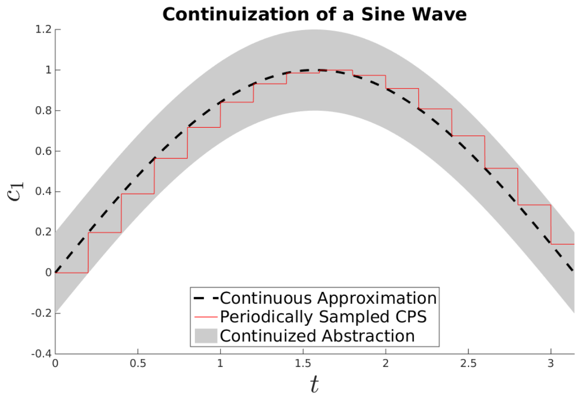


Fig. 3: The main idea behind the proposed continuization approach is that a nondeterministic continuous system contains the behaviors of a periodically sampled system.

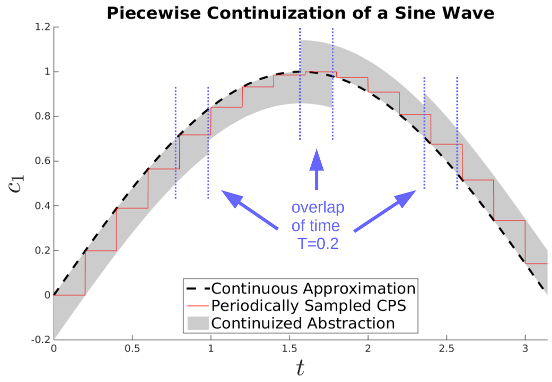


Fig. 4: The continuization approach as applied to four local analysis domains has an overlap of one period length between domains.

To provide some intuition on the construction of sampling deviation functions, we provide an simple illustrative example.

Example 2 (Sine Wave)**.** *Consider a system with a single cyber variable c_1 where the controller_update function is given by $\sin(t)$, and t is a clock (physical variable with $\dot{t} = 1$) ticking from 0 to π. The period of the cyber-variable is $T = 0.2$.*

The rate of change of *controller_update* (the derivative) is equal to $\cos(t)$, and the bound on cos in $[0, \pi]$ is $K = [-1, 1]$. Given this bound and the period of $T = 0.2$, each occurrence of c_i in the continuous approximation is replaced by $c_i + [-0.2, 0.2]$ in the continuized abstraction. A visual depiction of this is given in Fig. 3.

One nice property of the sampling deviation function constructed by Lemma 1 is that no matter how large the bounds are on the rate of change of the *controller_update* function, the sampling deviation function can be made arbitrarily small by choosing a small enough controller period T. This is because of the multiplication in the sampling deviation function by the interval $[-T, 0]$. Intuitively, this makes sense, since the continuous system is more closely approximated as we sample and actuate at a higher frequency. This is in contrast, however, to reachability analysis done directly on the sampled CPS models, where smaller periods lead to more discrete transitions, which lead to more error.

The width of the interval given by deviation function does affect the amount of overapproximation in the constructed model, and therefore it is desirable to have this function be as tight as possible. One way to improve the bound on the rate of change of the cyber variable is by considering smaller domains (time intervals). For example, we could take advantage of the time dependence of each sampling deviation function ω_i, and define corresponding sampling deviation functions within local analysis domains.

Lemma 2 (Sampling Deviation in Local Analysis Domains)**.** *For a cyber variable c_i with period T, given a sequence of interval bounds on the rate of change of the controller_update function, K_1, K_2, …, K_m, and an associated sequence of increasing and pointwise-intersecting time intervals (which we call local analysis domains) where the bounds are valid, $[t_0 = 0, t_1], [t_1, t_2], \ldots, [t_{m-1}, t_m]$, a sampling deviation function up to time t_m can be computed as:*

$$\omega_i(t) = [-T, 0] * [\min(\{K_j^{min} \mid t \in [t_{j-1}, t_j + T]\}),\\ \max(\{K_j^{max} \mid t \in [t_{j-1}, t_j + T]\})].$$

Proof. Notice the time intervals have the controller period T added to the upper time bound. This is because when a new time interval is entered in a trajectory, the sampled CPS could have taken the most-recent sample in either the current time interval, or in the previous one. The sampling deviation function, therefore, must account for both possibilities until T time has elapsed in the new interval. Other than this caveat, the proof follows that of Lemma 1, except that the analysis is done at each time interval. □

In the sine wave system from Example 2, we can apply this approach in four analysis domains (time intervals), $[0, \frac{\pi}{4}], [\frac{\pi}{4}, \frac{\pi}{2}], [\frac{\pi}{2}, \frac{3\pi}{4}], [\frac{3\pi}{4}, 1]$. Solving for the $\cos(t)$ (the derivative of $\sin(t)$) in these domains, we can come up with the associated interval bounds on $\dot{c}_1$ in the continuous approximation, $K_1 = [\frac{\sqrt{2}}{2}, 1], K_2 = [0, \frac{\sqrt{2}}{2}], K_3 = [-\frac{\sqrt{2}}{2}, 0], K_4 = [-1, -\frac{\sqrt{2}}{2}]$. Using the period $T = 0.2$, we then obtain the piecewise continuization of the system, shown in Fig. 4.

In Fig. 4, when the derivative bounds are positive, the difference between the sampled CPS and the continuous approximation is negative, which is why an interval of $[-T, 0]$ was used to bound the difference. Also, without the presence of the overlap between time domains, the continuized abstraction would be wrong immediately after time $\frac{\pi}{2}$. In this case, the new domain has a strictly negative derivative, but because the sample occurred before $\frac{\pi}{2}$, the bound from the previous domain must be used.

There are two considerations when applying continuization with local analysis domains. First, the result is only valid until the maximum time of the last analysis domain. If this time is finite, this means only bounded-time reachability can be computed. Second, there is a trade off between the accuracy of the computation and the number of domains considered. Continuization was originally used to eliminate large numbers of discrete transitions in a sampled CPS. Using local analysis domains, however, brings back discrete transitions, although now the number of transitions can be controlled by adjusting the number of domains. Using too many domains may lead to similar problems with tool performance as when we directly considered a sampled CPS model for reachability analysis. We could solve this problem by having sampling deviation functions that vary as continuous functions of time, although the way to create these is less clear, and left as possible future work.

IV. Automation in Hyst

Hyst [8] is a model transformation and translation tool for hybrid automaton models. Hyst performs both model *translation*, which converts between formats of different reachability tools, as well as model *transformations*, which serve to improve reachability computation results. The continuization approach described in the previous section has been implemented as a model transformation pass in Hyst, which permits easy application of the developed technique.

A. Transformation Pass

The implemented model transformation pass performs continuization starting given a continuous approximation of the system. The user provides (1) a target model file describing the hybrid automaton, (2) the controller period, T, (3) the name of the cyber variable of interest, c_i, (4) a sequence of m increasing times used to construct local analysis domains, (5) a corresponding sequence of m bloating terms, which will be described shortly, and (6) the name of the time variable (optional; only used if multiple local analysis domains are used to create transitions between them).

Given these inputs, the pass first simulates the continuous abstraction from the center of the initial states, in order to approximate the interval bounds on the rate of change of the derivative of c_i. For each time interval, the bound during that time is then expanded by the corresponding user-provided bloating term. We call the new intervals *candidate Lipschitz bounds* for the cyber variable's derivative. The candidate Lipschitz bounds are used as described in Lemma 2, along with the time domains, in order to produce the sampling deviation function $\omega_i(t)$.

The sampling deviation function consists of piecewise constant intervals. For each piece, a mode is created in the output hybrid automaton, with dynamics equal to the continuous approximation, except with every occurrence of c_i replaced by $c_i + \omega_i(t)$. Transitions are then added between the modes when the appropriate amount of time has elapsed.

The bound given by ω_i is only valid, however, if the candidate Lipschitz bounds are actually upper and lower bounds on the derivative of the cyber variable. This can happen because the bounds are constructed from a single simulation using the continuous approximation, whereas the reachable set of states considers all initial points as well as the expanded set of values for the cyber-variable in the dynamics, $c_i + \omega_i(t)$ instead of just c_i. To check if the bounds are respected, invariants and guards are added to the output hybrid automaton to check if the derivative exceeds the candidate Lipschitz bounds. If a violation occurs, a transition to an error state is taken, which is added as a forbidden location in the model. *In this way, performing reachability computation will not only give the set of states reachable by the continuized abstraction, but will also check that the candidate Lipschitz bounds are actual bounds on the derivative of the cyber variable*. If they are not, the transition to the error state will be detected when performing a reachability computation, and the transformation pass can be re-run with larger bloating terms, which will increase the size of the candidate Lipschitz bounds.

B. Example

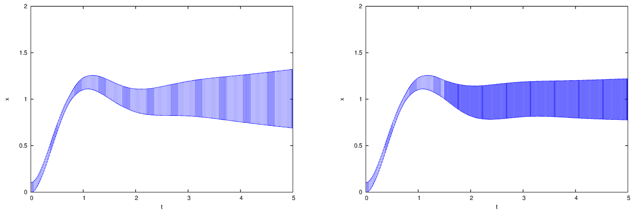
We apply the continuization approach in Hyst to the double-integrator system given in Example 1. The *controller_update* function in this case is $P * (1 - x) + D * -v$, with $P = 10$ and $D = 3$. The time derivative is $-10 * \dot{x} - 3 * \dot{v}$. After substituting in the derivatives ($\dot{x} = v$, $\dot{v} = a$), the derivative of a in the continuous abstraction is: $-10 * v - 3 * a$. The initial value of a is the value assigned when *controller_update* is evaluated at the initial states, $a := 10*(1-x)+3*-v$. The hybrid automaton of the continuous approximation shown in the appendix, in Fig. 9.

The pass implemented in Hyst performs a simulation of the system starting from the center of the initial set of states, in this case, at $x = 0.05$, $v = 0$, $a = 9.5$. The value of $\dot{a}$ in the simulation is observed to be in the interval $[-28.64, 5.27]$. This interval is then bloated by the provided bloating term, for which we consider ± 1, ± 2 and ± 4.

When running reachability with a bloating term of 1, Flow* immediately (at time 0) detects that the constructed error states are reachable, which means that the candidate Lipschitz bounds do not contain all the encountered values of $\dot{a}$. Computationally, we can show this to be the case. Initially, $x = [0, 0.1]$, which means the initial value of a is $[9, 10]$. The initial value of $\dot{a}$ is $-10*v-3*a = [-30, -27]$. The interval values of $\dot{a}$ in the simulation were $[-28.64, 5.27]$, which bloated by 1 give candidate Lipschitz bounds of $[-29.64, 6.27]$. The lower bound of the derivative of the cyber variable (-30) is initially outside of the candidate bounds, which was detected by the transition to the error state.

Using a bloating term of 2, the candidate Lipschitz bounds are $[-30.64, 7.27]$, which contain the above-computed initial values of $\dot{a}$. When performing reachability, however, at time 0.04 an error state is reached again. At this time, the reachable set contains a state where $a = 8.79$ and $v = 0.382$. In this case, the derivative $\dot{a} = -10 * v - 3 * a + \omega_i = -10 * 0.382 - 3 * 8.79 + [-0.45, 0.11]$ has a lower value of -30.66, which is below the candidate Lipschitz bound of -30.64.

When the larger bloating term of 4 is used, the candidate Lipschitz bound is respected by the reachable set, and Flow* does not reach the out-of-bounds error states. Thus, the reach



(a) Continuized System (b) Two Analysis Domains

Fig. 5: The response for the continuized periodically-controlled double-integrator system from Example 1 is significantly tighter than direct analysis (Fig. 2).

set of the continuized abstraction is a validated overapproximation of the reach set of the sampled CPS.

Recall, however, that directly computing the reach set of the sampled CPS, as shown in Fig. 2, resulted in a large exponential blow up in the size of the reachable set due to accumulation of overapproximation error. Even with a single analysis domain, the reachable set is significantly smaller, as shown in Fig. 5(a). Using multiple analysis domains, the reachable set can be further reduced. The hybrid automaton of the continuized system with two local analysis domains $[0, 1.5]$ and $[1.5, 5]$ is shown in the appendix in Fig. 10. The reach set of the response for this system is shown in Fig. 5(b). *Thus, the continuization method developed in this paper enables a more precise formal analysis of this system using hybrid automaton reachability tools.*

In terms of overhead, the runtime of the pass itself is small, taking about 100 ms. The reachability computation takes 0.9 seconds for the single-domain case, and about 1.3 seconds for the two-domain system, which is significantly faster than the 12 minutes needed for SpaceEx to produce Fig. 2(b).

V. Case Study

In this section, we apply the technique developed from Sec. III in order to perform reachability analysis of a hybrid system model of a yaw-damper for a 747 aircraft.

A. System and Controller Model

The model and controller we analyze in this case study are taken from the Control Systems Toolbox case studies in Matlab [13]. In brief, the system is a multiple-input multiple-output (MIMO) system that uses the aileron and rudder in order to reduce oscillations in the yaw and roll angle.

The analysis of the yaw damper is done on the system's aileron-to-bank angle impulse response. Three different systems are considered: (1) the original, undamped system, which experiences oscillations upon an impulse input, (2) the system with proportional compensator, which eliminates the oscillations but also over-stabilizes the spiral mode (a desired characteristic for the control), and (3) the system with a washout filter, which eliminates the oscillations but keeps the spiral mode.

We use this case study to evaluate the developed continuization technique so as to evaluate properties about the response of the final (washout filter) system. There are four physical variables in this system, sideslip angle (x_1), yaw rate (x_2), roll rate (x_3), and bank angle (x_4), represented by the column vector x. The two inputs u, are the rudder (u_1) and aileron (u_2). The outputs are the yaw rate and bank angle. The dynamics for the physical system are the standard linear time-invariant dynamics, $\dot{x} = Ax + Bu$ (the A and B matrices are provided in the in Sec. B of the appendix).

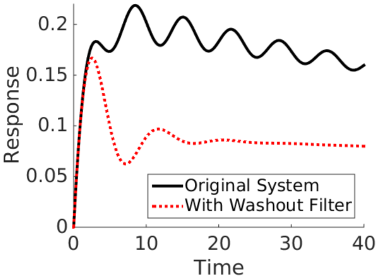


Fig. 6: The impulse response for the washout filter design of a yaw damper demonstrates the spiral mode in simulation.

This physical system is put into a feedback loop with a washout filter. The washout filter has a single variable, w, with dynamics $\dot{w} = x_2 - 0.2 * w$. The washout filter variable is combined with the yaw to produce an effect on the rudder input. That is, the washout filter adds to u_1 the value $2.34 * (x_2 - 0.2 * w)$.

A simulation of the aileron-to-bank angle impulse response from this system, with and without the washout filter, is given in Fig. 6. In particular, the two control properties of interest are a lack of oscillations (quick settling time), and the presence of the spiral mode. The spiral mode is a desirable flight characteristic demonstrated by the apparent[2] steady-state offset in the rudder-to-bank angle impulse response.

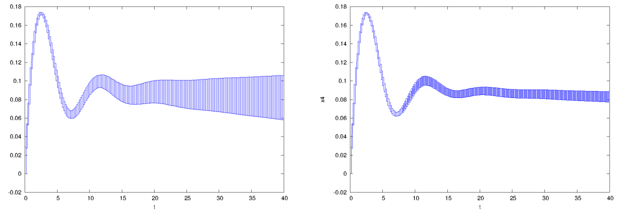
A property to check is that the aileron to bank angle impulse response remains around the simulated value of 0.08, between 20 and 40 seconds, and thus maintains the spiral mode without significant oscillation. We consider a controller running at 20 Hz ($T = 0.05$), using the implementation that samples and actuates when the real-time scheduled controller runs (Model 3 from Sec. II-B).

B. Reachability Analysis

Neither SpaceEx nor Flow* can effectively compute reachability on the periodically-actuated system model (Fig. 11 in the appendix). The reachable set of states explodes almost immediately, and neither tool can compute accurate time-bounded reachability for the required 40 seconds.

We apply the continuization approach developed in this paper by using the Hyst transformation pass on the continuous approximation of the model. First, we apply the technique over the whole time range. Initially, we try a small bloating term, and increase it until error states are no longer reachable during analysis. For the period parameter given to the pass, we use twice the control period, as this is needed to account for the

[2] The steady state is actually zero, but the convergence is very slow over hundreds of seconds.



(a) Reachability in Flow* of the Continuized Model

(b) Reachability with Local Domains and Halving Period

Fig. 7: Flow* can successfully compute reachability on the continuized model. When a smaller period and local analysis domain is used, the result is tighter.

maximum delay in sampling in Model 3, as discussed earlier in Sec. II-B. Flow* successfully computes reachability for the model, and confirms that the final bloating term (0.0007) was sufficiently large. The output plot is shown in Fig. 7(a).

Although the computation completes, which is an improvement over the direct computation, the set of states appears to be diverging slowly. The reachability result can be improved by using local analysis domains, or by reducing the controller period. To demonstrate this, we halve the controller period, and use two analysis domains. For time $[0, 8]$ we use a bloating term of 0.0004, and for time $[8, 40]$ we use 0.0003. Hyst creates the associated model file for Flow*, which we then use to compute reachability. Flow*, in about 5 seconds, confirms that the candidate domains are sufficient, and the resultant reachability plot is tighter than the previous one, as shown in Fig. 7(b). Furthermore, the spiral mode can be observed from the reachable set plot, along with the absence of oscillations in the time range $[20, 40]$.

VI. Related Work

In this paper, we have focused on controller analysis using hybrid automata reachability tools, although there are existing methods in control theory to design and analyze controllers. The design of a controller for a continuous-time system often occurs in continuous-time, and the controller is subsequently discretized[3] to be implemented in a software controller that operates periodically.

Continuous-Time Controller Design: There are many methods for control design in continuous-time. For example, a common strategy for linear time-invariant (LTI) systems is to design a stabilizing linear state-feedback controller of the form $u = Kx$ for a vector K [16]. Assuming the system is both controllable and observable, the strategy yields a new closed-loop system: $\dot{x} = Ax + Bu$ for $u = Kx$. After substituting this gives $\dot{x} = Ax + B(Kx)$ and then $\dot{x} = (A + BK)x$. This strategy is also known as pole placement [16]. Finding the vector K such that $(A + BK)$ is exponentially stable can be formulated in a variety of ways, such as by solving a linear matrix inequality (LMI) [17]. Linear quadratic regulator (LQR) design is another linear system design technique that also incorporates a cost function to yield an optimal controller [18]. LQR is used within the Linear Quadratic Gaussian (LQG) problem that robustly tolerates Gaussian additive noise inputs from disturbances. Other control design methods for linear systems are performed in the frequency domain, where pole and zero placement may also be performed to ensure stability and analyze performance criteria such as gain margins, phase margins, and use graphical tools like Nyquist diagrams and Bode plots. Design of controllers for nonlinear systems is challenging, but many approaches exist, such as linearizing and using gain-scheduled linear controllers, backstepping, feedback linearization, and many others [19].

Discretization of Continuous Controllers: Discretization typically consists of several steps. First, a sampling period must be selected at which measurements of the physical system are taken and made available to the software controller. Second, a control period must be selected to specify the rate at which control decisions are produced by the software controller and sent to actuators to influence the plant. Typically, these periods are selected in accordance with the speeds of the dynamics, and a common rule of thumb is to use the Nyquist frequency of the physical process to determine the minimum sampling period. The Nyquist frequency is twice the highest waveform frequency.

Given these periods, a discrete-time version of the plant can be constructed (using the sampling period) and a discrete-time version of the controller can be constructed (using the control period). Both discretizations are needed, as from the perspective of the controller, it will only receive state measurements of the plant at the points in time specified by the sampling period.

Discrete Controllers with Continuous Plants: While from the perspective of the software controller, the changes to the plant occur discretely, in reality, the plant evolves continuously according to differential equations. Controller performance with such constraints has been extensively investigated, and tools like JitterBug and TrueTime can characterize controller performance with real-time constraints and delays [20]. More recent works aid in synthesizing embedded software from hybrid systems models [21]. Giotto aids in this process of moving from control models to embedded real-time code [22].

Reachability: The elimination of large numbers of discrete transitions in hybrid automata was previously accomplished by continuization [7]. The earlier work was used to analyze properties about fast-switching electronic circuits. This work, in contrast, applied continuization to enable the analysis of fast-switching hybrid automata resulting from the periodic interactions with the real-time scheduler. We also considered using local analysis domains to construct the nondeterministic term, which was shown to increase the accuracy of the model.

Periodically Controller Hybrid Automata (PCHA) is one formalism for periodically-controlled embedded systems [23]. Automated analysis of PCHAs is possible only if the vector fields are polynomial, whereas, using the developed Hyst pass, continuization can be automatically applied to a broader

[3] In this paper, we only focus on the conversion from continuous-time to discrete-time, and do not consider full digitization [14], [15], for example, the conversion from continuous-time and continuous-state to discrete-time and discrete-state through quantization.

class of systems. Combinations of reachability tools and SMT solvers have been used to model both physical-world dynamics and software behavior [24]. A limitation of this approach is that cyber-variables are represented with intervals, and that only strictly-periodic systems can be analyzed (Model 1 from Sec. II-B).

VII. Conclusion

Analysis of large CPS using formal hybrid systems analysis techniques remains difficult. A challenge problem was recently proposed to the research community by Toyota on the verification of a powertrain control system [25]. Although initial progress has been made on simplified versions of the system [26], the full benchmark model presents four main challenges for verification tools: (1) controllers that periodically actuate the plant, (2) lookup tables to describe the system dynamics, (3) the presence of time delays in the model, and (4) large system scale.

In this paper, we addressed the first of these issues, by using continuization in order to soundly abstract the periodically-controlled dynamics. This permits initial analysis of these systems using reachability tools for hybrid automata. Without our approach, existing tools produce exponentially divergent reach sets on these models, and often fail before reaching the desired time bound. Since the accuracy of analysis depends on the tightness of the difference between the discrete system and continuized abstraction, a possible future improvement would be to compute these bounds in local domains based on the system state, in addition to time as proposed in this paper.

Acknowledgments

This material is based on research sponsored by the Air Force Research Laboratory under agreement number FA8750-15-1-0105, the Air Force Office of Scientific Research (AFOSR), in part through the Summer Faculty Fellowship Program (SFFP) and contract FA9550-15-1-0258. This material is based upon work supported by the National Science Foundation (NSF) under Grant Nos. 1464311 and 1527398. The U.S. government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of AFOSR, AFRL, or NSF.

References

[1] C. Liu and J. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *Journal of the Association for Computing Machinery*, vol. 20, no. 1, 1973.

[2] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine, "The algorithmic analysis of hybrid systems," *Theoretical Computer Science*, vol. 138, pp. 3–34, 1995.

[3] N. Lynch, R. Segala, and F. Vaandrager, "Hybrid i/o automata," *Inf. Comput.*, vol. 185, no. 1, pp. 105–157, Aug. 2003.

[4] S. Bogomolov, G. Frehse, M. Greitschus, R. Grosu, C. S. Pasareanu, A. Podelski, and T. Strump, "Assume-guarantee abstraction refinement meets hybrid systems," in *Hardware and Software: Verification and Testing - 10th International Haifa Verification Conference, HVC 2014, Haifa, Israel, November 18-20, 2014. Proceedings*, 2014, pp. 116–131.

[5] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, "Spaceex: Scalable verification of hybrid systems," in *Proc. 23rd International Conference on Computer Aided Verification (CAV)*, ser. LNCS. Springer, 2011.

[6] M. Althoff and B. H. Krogh, "Avoiding geometric intersection operations in reachability analysis of hybrid systems," in *Proceedings of the 15th ACM International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '12. New York, NY, USA: ACM, 2012, pp. 45–54.

[7] M. Althoff, S. Yaldiz, A. Rajhans, X. Li, B. Krogh, and L. Pileggi, "Formal verification of phase-locked loops using reachability analysis and continuization," in *Computer-Aided Design (ICCAD), 2011 IEEE/ACM International Conference on*, Nov 2011, pp. 659–666.

[8] S. Bak, S. Bogomolov, and T. T. Johnson, "HyST: A source transformation and translation tool for hybrid automaton models," in *18th International Conference on Hybrid Systems: Computation and Control (HSCC 2015)*. Seattle, Washington: ACM, Apr. 2015.

[9] T. A. Henzinger, P. W. Kopke, A. Puri, and P. Varaiya, "What's decidable about hybrid automata?" in *Journal of Computer and System Sciences*. ACM Press, 1995, pp. 373–382.

[10] X. Chen, E. Abraham, and S. Sankaranarayanan, "Taylor model flowpipe construction for non-linear hybrid systems," *IEEE Real-Time Systems Symposium*, vol. 0, pp. 183–192, 2012.

[11] ——, "Flow*: An analyzer for non-linear hybrid systems," in *International Conference on Computer Aided Verification (CAV)*, 2013.

[12] M. Althoff, A. Rajhans, B. H. Krogh, S. Yaldiz, X. Li, and L. Pileggi, "Formal verification of phase-locked loops using reachability analysis and continuization," *Commun. ACM*, vol. 56, no. 10, pp. 97–104, Oct. 2013.

[13] T. Mathworks, "Yaw damper design for a 747 jet aircraft," Control System Toolbox Examples, 2015. [Online]. Available: http://http://www.mathworks.com/help/control/examples/yaw-damper-design-for-a-747-jet-aircraft.html

[14] R. Brockett and D. Liberzon, "Quantized feedback stabilization of linear systems," *Automatic Control, IEEE Transactions on*, vol. 45, no. 7, pp. 1279–1289, Jul. 2000.

[15] T. T. Johnson, S. Mitra, and C. Langbort, "Stability of digitally interconnected linear systems," in *Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference (CDC ECC 2011)*, Orlando, Florida, USA, Dec. 2011, pp. 2687–2692.

[16] C. Chen, *Linear System Theory and Design*, 3rd ed. New York, NY: Oxford University Press, 1999.

[17] J. Löfberg, "Yalmip : A toolbox for modeling and optimization in MATLAB," in *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004. [Online]. Available: http://users.isy.liu.se/johanl/yalmip

[18] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal control*. John Wiley & Sons, 2012.

[19] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Upper Saddle River, NJ: Prentice Hall, 2002.

[20] A. Cervin, D. Henriksson, B. Lincoln, J. Eker, and K. Arzen, "How does control timing affect performance? analysis and simulation of timing using jitterbug and truetime," *Control Systems, IEEE*, vol. 23, no. 3, pp. 16–30, June 2003.

[21] M. Anand, S. Fischmeister, Y. Hur, J. Kim, and I. Lee, "Generating reliable code from hybrid-systems models," *Computers, IEEE Transactions on*, vol. 59, no. 9, pp. 1281–1294, Sep. 2010.

[22] T. Henzinger, C. Kirsch, M. Sanvido, and W. Pree, "From control models to real-time code using giotto," *Control Systems, IEEE*, vol. 23, no. 1, pp. 50–64, 2003.

[23] T. Wongpiromsarn, S. Mitra, A. Lamperski, and R. M. Murray, "Verification of periodically controlled hybrid systems: Application to an autonomous vehicle," *ACM Trans. Embed. Comput. Syst.*, vol. 11, no. S2, pp. 53:1–53:24, Aug. 2012.

[24] G. Simko and E. K. Jackson, "A bounded model checking tool for periodic sample-hold systems," in *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '14. New York, NY, USA: ACM, 2014, pp. 157–162.

[25] X. Jin, J. V. Deshmukh, J. Kapinski, K. Ueda, and K. Butts, "Powertrain control verification benchmark," in *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control*, ser. HSCC '14. New York, NY, USA: ACM, 2014, pp. 253–262.

[26] C. Fan, P. S. Duggirala, S. Mitra, and M. Viswanathan, "Progress on powertrain verification challenge with C2E2," in *ARCH '15: Proc. of the 2nd Workshop on Applied Verification for Continuous and Hybrid Systems*, April 2015.

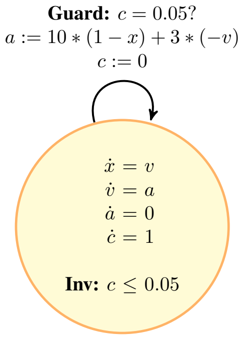


Fig. 8: Hybrid automaton model for sampled CPS of the double-integrator system in Example 1.

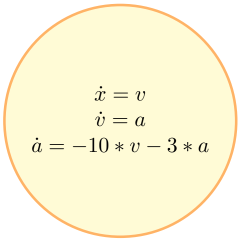


Fig. 9: Hybrid automaton model for continuous approximation of the double-integrator system in Example 1.

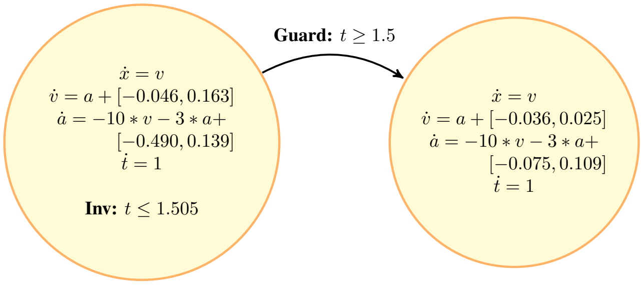


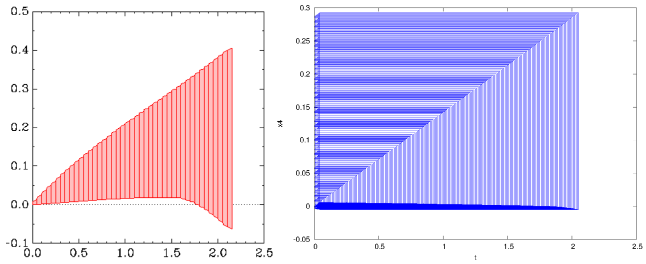
Fig. 10: Hybrid automaton model for continuized abstraction with two analysis domains (with error modes and transitions omitted) of the double-integrator system in Example 1.

Appendix

A. Double-Integrator Example

The hybrid automata for the double-integrator system (Example 1) are shown in Figs. 8, 9, and 10. In the continuous approximation and the continuized abstraction, the initial value of a is taken to be the value when *controller_update* is evaluated at the initial states, $a := 10*(1-x)+3*-v$.

The continuized abstraction shown in Fig. 10 is constructed from two time domains, $[0, 1.5]$ and $[1.5, 5]$, using a bloating term of 4 for each of the domains. The ranges of $\dot{a}$ in simulation for the two domains are $[-28.65, 5.27]$ and $[-0.97, 3.24]$, which give interval bounds of $K_1 = [-32.65, 9.27]$, and $K_2 = [-4.97, 7.24]$. With a period of $T = 0.005$, this gives interval values for ω of $[-0.046, 0.163]$ and $[-0.036, 0.025]$. The derivative $\dot{a}$ uses a value of -3 multiplied by these intervals due to the substitution of a by $a+\omega$ (since a is multiplied by -3 in the derivative). The derivative could have equivalently been written as $\dot{a} = -10*v-3*(a+[-0.036, 0.025])$.



(a) SpaceEx (b) Flow*

Fig. 11: Neither SpaceEx (left) nor Flow* (right) can directly compute reachability accurately on the yaw-damper model.

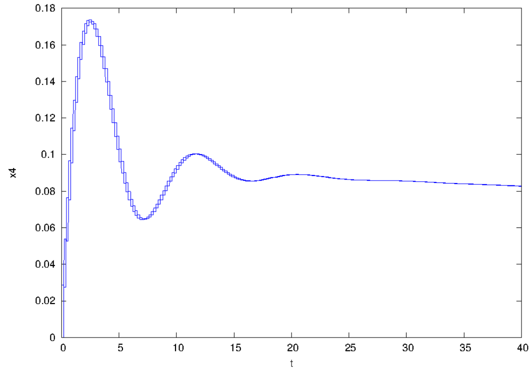


Fig. 12: The continuous approximation of the yaw-damper system demonstrates the spiral mode.

In Fig. 10, the error modes and transitions were not drawn. The guard conditions to enter an error mode in the first domain are $-10*v-3*a+0.139 \ge 9.27$ or $-10*v-3*a+-0.490 \le -32.65$. In the second domain, the guard conditions are $10*v-3*a+0.109 \ge 7.24$ or $-10*v-3*a+-0.075 \le -4.97$.

B. Yaw-Damper Example

The dynamics of the yaw-damper system from Sec. V are standard linear time-invariant dynamics, $\dot{x} = Ax + Bu$, with:

$$A = \begin{bmatrix} -0.0558 & -.9968 & 0.0802 & 0.0415 \\ 0.598 & -0.115 & -0.0318 & 0 \\ -3.05 & 0.388 & -0.4650 & 0 \\ 0 & 0.0805 & 1 & 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0.00729 & 0 \\ -0.475 & 0.00775 \\ 0.153 & 0.143 \\ 0 & 0 \end{bmatrix}.$$

Neither SpaceEx nor Flow* can compute reachability on the periodically-actuated system. The reachability plots produced by the reachability tools on the real-time actuated model (Model 3) are given in Fig. 11.

The continuous approximation of the system demonstrates the spiral mode and is close to the reach set for the periodically-actuated washout filter system. The plot for the continuous approximation is shown in Fig. 12. This is the system that is used as input to the Hyst continuization pass.

Scalable Static Hybridization Methods for Analysis of Nonlinear Systems[*]

Stanley Bak
Air Force Research Laboratory
Information Directorate, USA

Sergiy Bogomolov
IST Austria

Thomas A. Henzinger
IST Austria

Taylor T. Johnson
University of Texas at Arlington, USA

Pradyot Prakash
IIT Bombay, India

ABSTRACT

Hybridization methods enable the analysis of hybrid automata with complex, nonlinear dynamics through a sound abstraction process. Complex dynamics are converted to simpler ones with added noise, and then analysis is done using a reachability method for the simpler dynamics. Several such recent approaches advocate that only "dynamic" hybridization techniques—i.e., those where the dynamics are abstracted on-the-fly during a reachability computation—are effective. In this paper, we demonstrate this is not the case, and create static hybridization methods that are more scalable than earlier approaches.

The main insight in our approach is that quick, numeric simulations can be used to guide the process, eliminating the need for an exponential number of hybridization domains. Transitions between domains are generally time-triggered, avoiding accumulated error from geometric intersections. We enhance our static technique by combining time-triggered transitions with occasional space-triggered transitions, and demonstrate the benefits of the combined approach in what we call mixed-triggered hybridization. Finally, error modes are inserted to confirm that the reachable states stay within the hybridized regions.

The developed techniques can scale to higher dimensions than previous static approaches, while enabling the parallelization of the main performance bottleneck for many dynamic hybridization approaches: the nonlinear optimization required for sound dynamics abstraction. We implement our method as a model transformation pass in the HYST tool, and perform reachability analysis and evaluation using an unmodified version of SpaceEx on nonlinear models with up to six dimensions.

1. INTRODUCTION

A hybrid automaton [7] is an expressive mathematical model useful for describing complex dynamic processes involving both continuous and discrete states and their evolution. Efficient algorithms and analysis tools for linear and affine systems have recently emerged [24]. However, the behaviour of many real-world systems can only be modeled with nonlinear differential equations.

Hybridization methods attempt to address this issue, enabling the application of existing algorithms for simpler dynamics (such as constant or affine dynamics) on the analysis of hybrid automata with nonlinear differential equations. Alternative recent approaches for analyzing nonlinear systems include simulation-based verification [22] or using efficient representations such as Taylor models [17]. Most hybridization methods work by dividing the state space into a set of domains. In each domain, the nonlinear dynamics are then converted to simpler ones with added noise to account for the abstraction error within the domain. Hybridization is also known as *conservative approximation* [8], which illustrates that it is a sound (or conservative) abstraction. Hybridization has been used to verify properties for several types of systems, from analog/mixed-signal circuits [19] to autonomous satellite maneuvers in space [14, 31].

We classify existing hybridization approaches along two axes as shown in Table 1: static versus dynamic, and space-triggered versus time-triggered. *Static* hybridization approaches use a fixed partitioning, and can make use unmodified, off-the-shelf analysis tools. In contrast, *dynamic* methods exploit runtime information to perform hybridization, and therefore must be tightly integrated within an analysis tool. On the other axis, *space-triggered* techniques perform geometric intersections along hybridization domain boundaries. Time-triggered hybridization, on the other hand, avoids this operation by creating a series of overlapping domains, and switches between them at specific points in time.

Based on this classification, a gap exists in existing research: no methods exist that perform static, time-triggered hybridization. The main contribution of this paper is the investigation of this category, and demonstrating that such methods can overcome some of the drawbacks of existing hybridization methods. Notably, the new hybridization methods are more scalable than existing space-triggered approaches. Furthermore, the expensive dynamics abstraction step, which is generally a global optimization problem, is easily parallelizable, which is not the case in dynamic

[*]DISTRIBUTION A. Approved for public release; Distribution unlimited. (Approval AFRL PA #88ABW-2016-0181, 28 JAN 2016)

HSCC'16, April 12 - 14, 2016, Vienna, Austria

© 2016 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-3955-1/16/04...$15.00

DOI: http://dx.doi.org/10.1145/2883817.2883837

	Space-Triggered	Time-Triggered	Mixed-Triggered
Static	[8, 10, 29, 31]	this paper	this paper
Dynamic	[8, 9]	[1–3, 5, 20, 28]	none

Table 1: Breakdown of hybridization approaches into static versus dynamic, and space-triggered versus time-triggered, as well as combinations thereof (mixed-triggered).

approaches. We further enhance our static technique by combining time-triggered transitions with occasional space-triggered transitions, and demonstrate the benefits of the combined approach in what we call *mixed-triggered hybridization*.

The static mixed-triggered hybridization approach works by hybridizing only a part of the state space. We use quick numeric simulations to guide the partitioning process. In this way, we mitigate the problem of exponential growth in the number of partitions. In addition, we generally use time-triggered guards in the transitions between partitions. This prevents costly geometric intersection computations which typically add overapproximation error to the result. We ensure the soundness of the constructed abstraction by adding *error modes* to guarantee that the computed reachable states remain within the hybridized region (which is constructed from simulations that may be imprecise).

We implement the hybridization method described in this paper as a model transformation pass in the HYST source-to-source translation tool. Since it is a static approach, we can use unmodified reachability tools on the hybridized models. We create affine abstractions of nonlinear dynamics, and use to perform reachability analysis.

Contributions and Paper Organization. The main contribution of this paper is the development of the first static time-triggered and mixed-triggered hybridization methods. Of critical importance in the proposed approaches is the choice of hybridization parameters, and a second contribution is an algorithm which uses simulations to generate these values. This algorithm is implemented in the HYST [12] model transformation tool, which allows it to quickly be applied to new systems and with new simulation parameters. Finally, we validate our claims that the method is more scalable than existing static approaches by evaluating it on nonlinear models, including a six-dimensional water tank model, and then using an unmodified version of SpaceEx [13, 15, 24], which does not natively support nonlinear dynamics, to compute the set of reachable states.

This paper first reviews and classifies existing hybridization methods in Section 2. Section 3 then presents mathematical background and formalisms, which are used in Section 4 to give formal descriptions and correctness arguments for several hybrid automaton transformations. A simulation-based algorithm to create the hybridization parameters used by the transformations is described next in Section 5. Section 6 discusses the implementation in HYST and experimental reachability results in SpaceEx, followed by a conclusion in Section 7.

2. HYBRIDIZATION METHODS

In this section, we discuss and classify previous research on hybridization. Hybridiziation is the process of using simple dynamics with noise to create an abstraction of a system with more complicated, usually nonlinear, dynamics. This is done to enable the analysis of systems with the more complicated dynamics by methods which work exclusively on the simpler ones.

This process is typically targeted for flow-pipe construction methods, where the set of reachable states is iteratively computed or overapproximated at monotonically increasing instances in time, starting from an initial set of states. Computational approaches maintain some representation of the set of states at each time instances, which we informally refer to as the *currently-tracked* set of states.

Static Space-Triggered Hybridization. Early hybridization methods were both static and space-triggered [29]. In these approaches, the state space is partitioned using a (typically uniform) grid or mesh, and transitions are added along the partition boundaries, resulting in state-dependent switching. The advantage of this approach is that existing termination checking techniques can be used, which is particularly useful in the case of periodic systems where linearizing a bounded subset of the state-space is reasonable [31].

There are, however, three main drawbacks. First, static mesh construction is traditionally done without knowledge of the reachable states. Therefore, it requires computing the mesh over the entire state space (or bounded subset thereof), which scales exponentially with the number of continuous dimensions in the system. Second, the geometric intersections required by space-triggered approaches may introduce error during reachability computation [4, 17]. This is because such intersections can require tools to convert from precise internal representations such as zonotopes [25], support functions [27], or Taylor models [17], to simpler representations where intersection operations can be computed, such as polytopes [6]. After intersection, the simpler representation is then converted back to the internal representation for subsequent computation [26]. These conversions can result in overapproximations of the original currently-tracked set of states, adding error each time they are performed. Since hybridization can be done more accurately when domains are small, many intersection operations may be necessary and this can quickly lead to error explosion, as well as an explosion in the number of modes of the hybrid automaton. Third, the currently-tracked set of reachable states may leave a hybridization domain along multiple facets, requiring splitting and, later, possibly remerging the set of reachable states, which can be both computationally expensive and inaccurate [20].

Dynamic Space-Triggered Hybridization. In order to help increase scalability, methods were developed that perform hybridization during reachability analysis [8]. This results in dynamic methods where the domain construction and the abstraction process is performed on-the-fly and only on states that are reachable [9]. Although dynamic space-triggered methods scale better into higher dimensions, they still suffer from the other two problems mentioned above: error accumulation due to many geometric intersections, and the splitting of the currently-tracked set of states along multiple facets.

Dynamic Time-Triggered Hybridization. To address the other two drawbacks, dynamic time-triggered approaches were developed [5, 20, 28]. These methods avoid geometric intersections by choosing hybridization domains around the currently-tracked set of states. As time is advanced, the hybridization domains are updated to be near the new position of the currently-tracked set of states, without requiring an intersection operation. This can be done at each step [28], or whenever the currently-tracked set of states leaves the hybridization domain [20]. This can be viewed as the mode of the abstract hybrid automaton changing at specific instances in time to a mode with new dynamics, which corresponds to a time-triggered transition.

Although dynamic time-triggered methods perform well, they also suffer from certain drawbacks. The most important drawback is that, in the earlier static approaches, performing the dynamics abstraction step was an embarrassingly parallel problem, so parallelism could be leveraged to reduce total runtime (or equivalently, increase precision for a fixed runtime). In dynamic methods, the bounds of each new abstraction domain depend on the set of reachable states in the previous domain, forcing this expensive step to be performed serially. For example, abstracting nonlinear dynamics using polynomial differential inclusions can yield an accurate hybridization, but it requires bounding the Lagrange remainder of the dynamics' Taylor expansion [1]. In previous work, this step was reported to take 1121 out of 1180 seconds on a nine-dimensional biological aging model (about 95% of the runtime), and 1155 out of 1296 seconds on hybrid variant of the same model (about 89%), although it was mentioned that some implementation optimizations were possible [1]. Some parallelization of reachability computation was considered to enable online reachability of car manoeuvres [2,3]. However, the crucial step of dynamics abstraction (computing the linearization errors) was still performed serially because the overapproximation of the Lagrange remainders of the Taylor expansions of the dynamics at each step was based on the Lagrange remainders at the previous step. This serial step dominated the reported runtime of the technique.

A second drawback of time-triggered approaches is that, if the currently-tracked set of states becomes large (which can be a property of the system regardless of the method used), the domains over which dynamics abstraction is performed also become large. This, in turn, increases the dynamics approximation error that must be added to the simpler dynamics to result in a sound abstraction, increasing error in the overapproximation of the set of reachable states. This can be overcome by splitting the set of reachable states [21], although this may yield an exponential number of sets that need to be tracked, and possibly redundant computation. This problem can be partially mitigated through extra tracking to perform cancellation of redundant sets of reachable states, which requires (expensive and error-introducing) intersection operations on the internal representations [5]. Space-triggered approaches do not suffer from this problem. In fact, introducing *occasional* artificial space-triggered transitions can serve to reduce the size and complexity of the currently-tracked set of reachable states [11].

Novel Hybridization Approaches. A classification of existing hybridization research is shown in Table 1. A research gap is noticeable in the static time-triggered category. This paper attempts to fill this gap by developing, to the best of the authors' knowledge, the first static time-triggered hybridization method. The approach is static, and therefore can perform the bottleneck step of dynamics abstraction in a parallel fashion. Since the approach is time-triggered, it can scale to larger numbers of dimensions while avoiding the accumulation of intersection error. Additionally, as the method is static and modifies the model directly, it can work with unmodified reachability tools, yielding immediate benefit of its application using the latest reachability methods.

There are also no fundamental reasons why a method could not use both time-triggered and space-triggered transitions during analysis. We develop such a *mixed-triggered* hybridization approach, which generally uses time-triggered transitions, but occasionally performs a state-triggered transition to attempt to reduce the size and complexity of the currently-tracked set of states. In our review of existing research, no such approaches currently exist.

Other Hybridization Factors. Research in hybridization also explores other aspects that are important, but less critical to the methods developed in this paper. One choice when performing hybridization is the shape of space-triggered domains. Rectangular domains are simple to reason about, although manual region selection [29], simplexes [9, 21, 31], and nonuniform meshes [8,10,31] have been considered. The sound and tight abstraction of dynamics within each domain is critical to control error when performing hybridization. The main reason to consider alternative domains is in order to reduce this error. For general nonlinear dynamics, this often requires solving constrained nonlinear optimization problems, which can be impossible in theory and expensive in practice. For rectangular domains, interval analysis [30] can be used to provide guaranteed bounds for this problem. For other types of domains, the success of the method depends on the system being analyzed. For example, to perform the nonlinear optimization step for simplicial domains, one can use knowledge of the system's Lipschitz constant (which will be sound but inaccurate), or compute bounds on the second partial derivatives (the elements of the Hessian matrix) [8, 9, 21]. In general, this is a nonlinear optimization problem with linear constraints, but for specific cases it can be efficiently solved. For example, for quadratic dynamics [20, 21], the Hessian matrix is constant. The choice of domains is not critical to the methods being developed in this paper, so for simplicity, we considered rectangular domains.

A second choice when performing hybridization is the type of 'simpler' dynamics. Choices range from constant bounds [16, 29, 31, 32], linear and affine bounds [9, 21, 31], to polynomial bounds [1, 18]. In this paper, we target an unmodified implementation of the SpaceEx tool [24], and therefore simplify from nonlinear dynamics to affine dynamics.

3. PRELIMINARIES

In order to define and justify the soundness of the model transformation steps used in our approach, we need to first precisely define the syntax and semantics of hybrid automata.

Definition 1. A *hybrid automaton* $\mathcal{H}$ is defined by a tuple $\mathcal{H} \triangleq (\mathit{Modes}, \mathit{Var}, \mathit{Init}, \mathit{Flow}, \mathit{Trans}, \mathit{Inv})$, where: (a) *Modes* is a finite set of modes. (b) $\mathit{Var} = \{x_1, \ldots, x_n\}$ is a set of real-valued variables. (c) $\mathit{Init}(m) \subseteq \mathbb{R}^n$ is the set of initial values for $x_1, \ldots, x_n$ for each mode $m \in \mathit{Modes}$. (d) For each $m \in$ *Modes*, the flow relation *Flow*(m) is a relation over the variables in x and their derivatives $\dot{x} = f_m(x)$, where $x(t) \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to 2^{\mathbb{R}^n}$, i.e., differential inclusions are allowed. (e) *Trans* is a set of discrete transitions $t = (m, g, \upsilon, m')$, where m and m' are the source and the target modes, g is the guard of t, and υ is the update of t. (f) *Inv*$(m) \subseteq \mathbb{R}^n$ is an invariant for each mode $m \in$ *Modes*.

For a time interval T, we define a *trajectory* of $\mathcal{H}$ from state $s = (m, \mathbf{x})$ to state $s' = (m', \mathbf{x}')$ as a tuple $(L, \mathbf{X})$. In this tuple, the function $L : T \to Modes$ and $\mathbf{X} : T \to \mathbb{R}^n$ are functions that define for each time point in T the mode and values of the continuous variables, respectively.

A state s' is *reachable from a state* s if there exists a trajectory starting with s and ending with s'. A state s' is *reachable* if s' is reachable from a state s where s is an initial state. We denote the set of states reachable from the set X in mode m by $\mathrm{Reach}_{\mathcal{H}}(m, X)$. $\mathrm{Reach}(\mathcal{H})$ of $\mathcal{H}$ is defined as the set of states that are reachable from the set of initial states. We use $\mathrm{Reach}^c_{\mathcal{H}}(m, X)$ and $\mathrm{Reach}^c(\mathcal{H})$ to denote the versions of the these operators that return only the *continuous* part of the computed state space. We refer to $\mathrm{Reach}^c(\mathcal{H})$ as the *continuous reachable state space* of $\mathcal{H}$. We denote the projection of the set $R \subseteq \mathbb{R}^n$ over variables *Var* to the subset $Var' \subseteq Var$ by $R \downarrow_{Var'}$. Throughout the paper, we always refer to time-bounded reachability, i.e., we consider trajectories which evolve up to the time horizon T_{max}. In order to simplify notations, we implicitly take this assumption for granted in our reasoning. Finally, given a mode m of the automaton $\mathcal{H}$, we refer to the set of *outgoing* transitions as $Trans_{\mathcal{H}}(m)$.

4. TRANSFORMATIONS

We are interested in methods to compute an overapproximation of the time-bounded set of reachable states, which produce tight overapproximations, yet are feasible from the computational point of view. The proposed approaches rely on several hybrid automaton transformations. A source-to-source *transformation* takes as input a hybrid automaton $\mathcal{H}$, a mode $m \in$ *Modes*,[1] possibly some additional parameters, and returns as output another hybrid automaton $\theta(\mathcal{H})$. The four described transformations are (1) *time-triggered splitting*, (2) *space-triggered splitting*, (3) *domain contraction*, and (4) *dynamics abstraction*. In time-triggered splitting, a given mode of $\mathcal{H}$ is split into possibly multiple modes via a time-triggered splitting of the modes. Similarly, in space-triggered splitting, a mode is split by augmenting the mode invariant with a constraint induced by a *space trigger function*. Domain contraction adds auxiliary invariants called *contraction domains* to a mode by intersecting them with the existing invariants of the mode. Dynamics abstraction overapproximates the dynamics in a mode of the automaton, which in this paper, abstracts nonlinear differential equations by linear differential inclusions, in particular a linear differential equation with an additive set-valued (interval vector) input.

As hybridization of the *continuous dynamics* of hybrid automata is the most challenging part of the hybridization process, we focus on the continuous dynamics of hybrid systems in the rest of the paper and assume that an input hybrid automaton has only *one* mode. Our approach *overapproximates* the behavior of the original system by a hybrid automata consisting of *multiple* modes. Therefore, only reachable *continuous states* are relevant for the soundness of the transformations. This fact allows us to to conclude that the inclusion of the original *continuous* reachable state space into the transformed one is enough to show soundness of our transformations. Note, however, that although the input hybrid automaton for the whole hybridization approach is assumed to be a singleton, our transformations are defined in terms of general hybrid automata.

In this section, each of these four transformations is precisely defined. After, these will be combined in order to perform static time-triggered and mixed-triggered hybridization.

4.1 Time-Triggered Splitting

The time-triggered splitting transformation, informally, separates the handling of system behavior in the first τ time units, and the rest of the trajectory up to the time horizon. In order to achieve this goal, the transformation splits a given mode of a hybrid automaton into two and imposes constraints that guarantee that the system dwells in the first mode for τ time units and proceeds to the second one once the time threshold has been reached.

Definition 2. A *time-triggered splitting* is a transformation θ_{tt} of a hybrid automaton $\mathcal{H}$, that takes as input an automaton $\mathcal{H}$, a mode $m \in$ *Modes* that has no outgoing transitions[2], and a real positive time τ, a *time-trigger threshold.* The hybrid automaton $\mathcal{H}_{tt} \triangleq \theta_{tt}(\mathcal{H})$ is defined as: (a) $Modes_{\mathcal{H}_{tt}} \triangleq Modes_{\mathcal{H}} \cup \{m_{tt}\}$, where m_{tt} is a fresh (i.e., unique) mode name, (b) $Var_{\mathcal{H}_{tt}} \triangleq Var_{\mathcal{H}} \cup \{t\}$, where t is known as the *time-trigger* variable and is fresh, i.e., assume without loss of generality that t is a unique variable name,[3] (c) the initial states are copied; in addition, if $Init_{\mathcal{H}}(m)$ is not the empty set (i.e., m is an initial mode), then $Init_{\mathcal{H}_{tt}}(m) \triangleq Init_{\mathcal{H}}(m) \wedge t = \tau$, and otherwise $Init_{\mathcal{H}_{tt}}(m) \triangleq Init_{\mathcal{H}}(m)$; $Init_{\mathcal{H}_{tt}}(m_{tt}) \triangleq \emptyset$, (d) the flows are copied, and $Flow_{\mathcal{H}_{tt}}(m_{tt}) \triangleq Flow_{\mathcal{H}}(m)$, so mode m_{tt} copies the original dynamics of m, and in m, $\dot{t} = -1$, and in all modes other than m, $\dot{t} = 0$, (e) the transitions are copied; in addition, $Trans_{\mathcal{H}_{tt}}(m_{tt}) \triangleq Trans_{\mathcal{H}}(m)$, with an additional transition created from m to m_{tt} with the guard $t = 0$; moreover, every incoming transition to m has the reset $t := \tau$ added, (f) the invariants are copied; in addition $t \geq 0$ is added to $Inv_{\mathcal{H}_{tt}}(m)$ and $Inv_{\mathcal{H}_{tt}}(m_{tt}) \triangleq Inv_{\mathcal{H}}(m)$ (m_{tt} copied the original invariant of m).

Figure 1 illustrates the time-triggered splitting for a single mode. A *time-triggered transition* corresponds to any transition with guard $t = 0$ taken when the time-trigger variable

[1] For simplicity of presentation, each transformation is defined for a given mode of the hybrid automaton $\mathcal{H}$, and their application to multiple modes of $\mathcal{H}$ is straightforward by iterating over each element of *Modes*.

[2] In order to make the presentation of our transformation clearer, we consider a mode with no outgoing transitions. Our construction can be easily generalized to also accommodate this feature.

[3] If the time-triggered splitting transformation θ_{tt} is applied to an automaton multiple times, the time-trigger variable may be reused in each splitting, as it needs only to be fresh on the first application of the transformation. This optimization is done in our implementation.

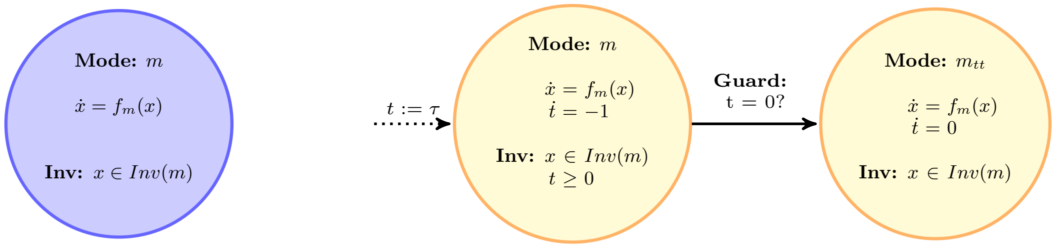


Figure 1: The time-triggered splitting transformation applied to the original automaton (left, blue) produces the output automaton (right, yellow). An additional time-trigger variable t is added that counts down to zero from an initial time τ.

$t = 0$. In contrast to general guards, the reachability along time-triggered transitions can be computed computationally efficient as many reachability algorithms automatically capture *time dependencies* as part of their workflow. For example, the STC scenario [23] of the hybrid model checker SpaceEx computes *time-dependent* piecewise-linear approximations of the support functions evolution.

The following lemma connects the time-triggered splitting transformation with the original hybrid automaton.

Lemma 4.1. *Let $\mathcal{H}$ be a hybrid automaton with a set of continuous variables Var, $m \in$ Modes be a mode without outgoing transitions, and $\tau \in \mathbb{R}_{>0}$ be a time-trigger threshold. Then it holds that* $\mathrm{Reach}^c(\mathcal{H}) \subseteq \mathrm{Reach}^c(\theta_{tt}(\mathcal{H})) \downarrow_{Var}$.

Here, we note that we need to project away the auxiliary variable t in order to ensure that the sets of reachable states of $\mathcal{H}$ and $\theta_{tt}(\mathcal{H})$ can be compared.

4.2 Space-Triggered Splitting

Space-triggered splitting, similar to time-triggered splitting, breaks a given mode into several modes. However, in contrast to the time-triggered transformation, it uses a *space-trigger function* to define criteria for mode splitting.

Definition 3. A *space-triggered splitting* is a transformation θ_{st} of a hybrid automaton $\mathcal{H}$, that takes as input an automaton $\mathcal{H}$, a mode $m \in Modes$ that has no outgoing transitions, and a function $\pi : \mathbb{R}^n \to \mathbb{R}$ called the *space-trigger* function. The function π must satisfy the condition that upon entering mode m, $\pi(x) \geq 0$, where x is the current state. This means that if m is an initial mode, for all states $x \in Init(m)$, $\pi(x) \geq 0$. The hybrid automaton $\mathcal{H}_{st} \triangleq \theta_{st}(\mathcal{H})$ defined as: (a) $Modes_{\mathcal{H}_{st}} \triangleq Modes_{\mathcal{H}} \cup \{m_{st}\}$, where m_{st} is a fresh (i.e., unique) mode name, (b) $Var_{\mathcal{H}_{st}} \triangleq Var_{\mathcal{H}}$, (c) the initial states are copied; $Init_{\mathcal{H}_{st}}(m_{st}) \triangleq \emptyset$, (d) the flows are copied; in addition, $Flow_{\mathcal{H}_{st}}(m_{st}) \triangleq Flow_{\mathcal{H}}(m)$, (e) the transitions are copied; in addition, $Trans_{\mathcal{H}_{st}}(m_{st}) \triangleq Trans_{\mathcal{H}}(m)$; moreover, an additional transition created from m to m_{st} with the guard $\pi(x) = 0$, and (f) the invariants are copied, with $\pi(x) \geq 0$ added to $Inv_{\mathcal{H}_{st}}(m)$ and $Inv_{\mathcal{H}_{st}}(m_{st}) \triangleq Inv_{\mathcal{H}}(m)$ (m_{st} copied the original invariant of m).

The space-triggered splitting transformation adapts the idea of pseudo-invariants [11] to the hybridization setting. In our setting, a space-trigger function π basically plays a role of a pseudo-invariant.

The resulting automaton overapproximates the continuous reachable state space of the original one which is formally stated in the following lemma.

Lemma 4.2. *Let $\mathcal{H}$ be a hybrid automaton, $m \in Modes$ be a mode without outgoing transitions, and $\pi : \mathbb{R}^n \to \mathbb{R}$ be a function satisfying the assumptions in Definition 3. Then* $\mathrm{Reach}^c(\mathcal{H}) \subseteq \mathrm{Reach}^c(\theta_{st}(\mathcal{H}))$.

4.3 Domain Contraction

Domain contraction adds auxiliary invariants known as *contraction domains* that should contain the set of reachable states. Given a set D and a mode m of a hybrid automaton $\mathcal{H}$ where $\dot{x} = f_m(x)$, if $\mathrm{Reach}_{\mathcal{H}}(m, X) \subseteq D$ for $X \subseteq Inv(m)$, i.e. the set of reachable states from mode m starting from a subset $X \subseteq Inv(m)$ is contained in D, then D may safely be added as an invariant of m. Of course, the set of reachable states is not available and is what is being computed or approximated, so error modes known as *domain contraction error modes* (DCEMs) are used to maintain soundness if the system leaves the states represented by these auxiliary invariants.

Definition 4. A *domain contraction* is a transformation θ_{dc} of a hybrid automaton $\mathcal{H}$, that takes as input an automaton $\mathcal{H}$, a mode $m \in Modes$, and a set $D \subseteq \mathbb{R}^n$ called the *contraction domain* auxiliary invariant.

The transformed hybrid automaton $\mathcal{H}_{dc} \triangleq \theta_{dc}(\mathcal{H})$ is defined as: (a) $Modes_{\mathcal{H}_{dc}} \triangleq Modes_{\mathcal{H}} \cup \{err\}$, the modes are the copied, with a new *domain contraction error mode* (DCEM) err added, (b) $Var_{\mathcal{H}_{dc}} \triangleq Var_{\mathcal{H}}$, (c) the initial states are copied; additionally, if m is an initial mode, and $Init(m)$ is not entirely contained in D, then add the err DCEM to the initial states; in this way, we capture a degenerate case if the initial set has states outside of the contraction domain. (d) the flows are copied; additionally, $Flow_{\mathcal{H}_{dc}}(err)$ of the form $\dot{x} = 0$ are added, (e) the transitions are copied, with additional transformations of the following form: given an incoming transition $d = (n, g, \upsilon, m)$ to mode m in $\mathcal{H}$, (1) augment the guard of the transition d with $x \in D$, and (2) add an additional transition $d' = (n, g \wedge x \in cl(\bar{D}), err)$ with an extra condition $x \in cl(\bar{D})$ on the guard and leading to the DCEM err, where $\bar{D}$ denotes the complement of D and $cl(\cdot)$ stands for topological closure and (3) add an additional transition $d'' = (m, x \in cl(\bar{D}), err)$, (f) the invariants are copied, except for the invariant $Inv_{\mathcal{H}_{dc}}(m) \triangleq Inv_{\mathcal{H}}(m)$ $\cap$ $x \in D$.

A visualization of the domain contraction transformation is given in Figure 2.

The conditions to enter a DCEM together ensure that regardless of the choice of the contraction domain, if the DCEM err is not reached, then the overapproximation of the reachable states is sound. Additionally, the condition that the dynamics are zero in the DCEM err ensures that during

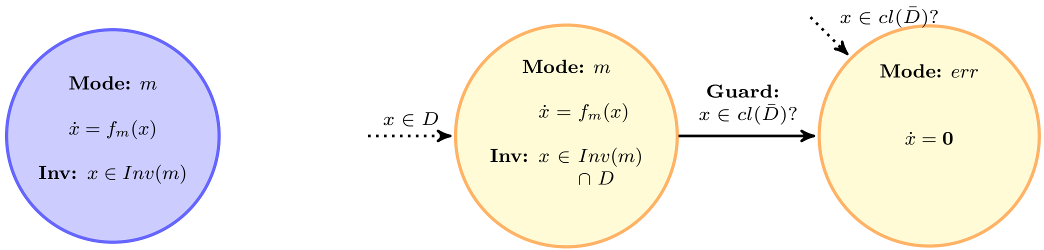


Figure 2: The domain contraction transformation applied to the original automaton (left, blue) produces the output automation (right, yellow). The contraction domain D is added to the invariant, with DCEM *err* inserted to detect if the reachable set of states leaves D.

a reachability computation, the exploration of the *err* will terminate and be a dead-end in the exploration of the state-space. Note that the notion of topological closure is required to ensure that the intersection of guard and invariant is non-empty.

LEMMA 4.3. *Let $\mathcal{H}$ be a hybrid automaton, $m \in$ Modes be a mode, and $D \subseteq \mathbb{R}^n$ be a contraction domain. Then, if no DCEM is reachable,* $\text{Reach}^{\text{c}}(\mathcal{H}) \subseteq \text{Reach}^{\text{c}}(\theta_{dc}(\mathcal{H}))$.

The contraction domain auxiliary invariants may be arbitrary and may be determined using any method, so they may not actually contain the set of reachable states. To maintain soundness, the DCEMs are added such that if the contraction domains *do not* contain the set of reachable states, transitions to the DCEMs *may* be taken.[4] If no DCEMs are reached, then the domain contraction transformation is sound, but otherwise, if a DCEM is reached, the resultant set of set of reachable states may not be subset of the original automaton's set of reachable states. If it is known that the set of reachable states will not leave the contraction domain by some other analysis, then the DCEMs are not necessary and the invariants may simply be augmented (conjuncted) with the contraction domain. In summary, if the contraction domains do not contain the set of reachable states for a given mode, then a state with a mode equal to the DCEM will be reached.

4.4 Dynamics Abstraction

Continuous dynamics are abstracted by transforming the flows of the original hybrid automaton into flows with increased nondeterminism. In this paper, nonlinear differential inclusions are overapproximated using linear differential inclusions, specifically linear ODEs with an additive set-valued input.

Definition 5. A *dynamics abstraction* is a transformation θ_{da} of a hybrid automaton $\mathcal{H}$, that takes as input an automaton $\mathcal{H}$, a mode $m \in Modes$, and a set-valued function $g : \mathbb{R}^n \to 2^{\mathbb{R}^n}$ called the *abstract dynamics*, where, for the flow $\dot{x} = f_m(x)$ of mode m with invariant $Inv(m)$, $g_m(x)$ is such that $\forall x \in Inv(m)$: $f_m(x) \subseteq g_m(x)$. The hybrid automaton $\mathcal{H}_{da} \triangleq \theta_{da}(\mathcal{H})$ is defined as: (a) $Modes_{\mathcal{H}_{da}} \triangleq Modes_{\mathcal{H}}$, (b) $Var_{\mathcal{H}_{da}} \triangleq Var_{\mathcal{H}}$, (c) the initial states are copied, (d) the flows are copied, except for $Flow_{\mathcal{H}_{da}}(m)$ which is set to $\dot{x} = g(x)$, (e) the transitions are copied, (f) the invariants are copied.

Similarly to other transformations we have considered, we formulate a lemma relating the original and transformed systems.

LEMMA 4.4. *Let $\mathcal{H}$ be a hybrid automaton, $m \in$ Modes be a mode, $g : \mathbb{R}^n \to 2^{\mathbb{R}}$ be a set-valued function satisfying the assumptions in Definition 5. Then it holds that* $\text{Reach}^{\text{c}}(\mathcal{H}) \subseteq \text{Reach}^{\text{c}}(\theta_{da}(\mathcal{H}))$.

5. MIXED-TRIGGERED HYBRIDIZATION

Now we present the central result of the paper, a static mixed-triggered hybridization that combines the four transformations we have introduced.

Definition 6. A static *mixed-triggered hybridization* is a transformation θ_{mt} of a hybrid automaton $\mathcal{H}$ and has the following input:

- a single-mode automaton $\mathcal{H}$,
- a list of splitting elements $E_1 \ldots . E_{n-1}$, where each element E_i is either a real number to be used for time-triggered splitting, or a π function to be used for space-triggered splitting (list 1),
- $D_1, \ldots, D_n$ are the contraction domains (sets) for each new location (list 2), and
- $g_1, \ldots, g_n$ are the dynamics abstraction functions for each location (list 3).

The mixed-triggered hybridization transformation consists of the following three steps:

- Apply either time-triggered splitting or space-triggered splitting based on the list $E_1 \ldots . E_{n-1}$. We apply each transformation to the most-recently constructed mode, which has no outgoing transitions. The result of this step is a chain of modes.
- For each mode in the chain, apply N domain contractions based on the list $D_1, \ldots, D_n$.
- For each mode in the chain, apply N dynamics abstractions based on the list $g_1, \ldots, g_n$.

If the list of splitting elements (list 1) contains only time-triggered splitting elements (and no space-triggered splitting elements), then it is a *static time-triggered hybridization.*

The following theorem establishes the soundness of the mixed-triggered hybridization.

THEOREM 5.1. *For hybrid automaton $\mathcal{H}$, if no DCEM are reachable, then the continuous reachable state space of the mixed-triggered transformation $\theta_{mt}(\mathcal{H})$ overapproximates the continuous reachable state space of the original automaton:* $\text{Reach}^{\text{c}}(\mathcal{H}) \subseteq \text{Reach}^{\text{c}}(\theta_{mt}(\mathcal{H}))$.

[4] Since the semantics of hybrid automata defined do not support urgency or *must* transitions, we exploit the fact that the reachability computation explores all paths to ensure soundness.

PROOF. The proof follows by a straight-forward application of Lemmas 4.1, 4.2, 4.3, and 4.4. □

We observe that the mixed-triggered hybridization approach contains a number of parameters which must be carefully chosen in order to guarantee a sound abstraction, which is ensured when no error modes (DCEMs) are reachable. If the contraction domains are too small, then the set of reachable states will exit the domain and the DCEM will be reached. If the contraction domains are too large, then the dynamics abstraction will be a large overapproximation, and the set of reachable states will become both large and inaccurate. In modes copied during time-triggered splitting, whenever the time-triggered variable t reaches zero, the set of reachable states at each mode must be contained in the domains (invariants) of both the source and target locations. Space-triggered splitting requires as input the π functions which determine the splitting structure.

In the following, we describe an approach to generate the parameters for proposed hybridization approach in a way that will satisfy the above requirements. Again, the approach is described assuming a single-location hybrid automaton, where the initial set of states is a rectangle, although generalizations are not difficult.

5.1 Parameter Selection Algorithm

In order to construct the three lists to be used as hybridization parameters, an algorithm is proposed which uses numerical simulations. The proposed approach has its own user-provided parameters:

- T is the maximum time,
- $\mathcal{S}$ a simulation strategy, one of {POINT, STAR, STARCORNERS}
- δ_{tt} is the simulation time in a time-triggered transformation step,
- n_{pi} is the number of space-triggered transformation steps to use,
- δ_{pi} is the maximum simulation time when performing a space-triggered transformation step,
- ϵ is a bloating term to account for the difference between the simulated points the set of reachable states.

The algorithm first selects a finite set of simulation points sampled from the initial set of states. If $\mathcal{S}$ is POINT, only the center of the initial rectangle is used. If $\mathcal{S}$ is STAR, the center is used, as well as the center of every face of the rectangle, $1 + 2n$ points, where n is the number of variables. If $\mathcal{S}$ is STARCORNERS, the center is used, as well as the centers of every face, as well as the corners of the initial rectangle, $1 + 2n + 2^n$ points. Selecting more points may permit a smaller ϵ, but since the number of points is exponential, the STARCORNERS strategy may not always be practical. The collection of points are stored in a variable, `sims`.

The algorithm proceeds in iterations, at each iteration doing either a *space-triggered step*, or a *time-triggered step*. The three parameter lists (the output) are initially empty. A current time variable `ct`, initially zero, is maintained which tracks the amount of time elapsed during time-triggered steps (space-triggered steps do not add to `ct`). A second variable `next_st` tracks the time at which to insert the next space-triggered value. If $n_{pi} > 0$, `next_st` is initialized to 0, otherwise it is set to ∞.

At each iteration, if the current time `ct` variable is greater than or equal to next space-triggered time variable `next_st`, a space-triggered step is *attempted* and `next_st` is increased by $\frac{T}{n_{pi}}$. Otherwise, a time-triggered transition is performed and `ct` is increased by δ_{tt}. The process completes when `ct` exceeds the maximum time T.

A **time-triggered step** adds δ_{tt} to output list 1. Then, it computes the bounding box of `sims`, bloats it by ϵ, and stores it in `start`. Each point in `sims` is numerically simulated for δ_{tt} time. The bounding box of `sims` is computed again, bloated by ϵ, and stored in `end`. The bounding box of `start` and `end` is then computed, and put into output list 2 (contraction domains).

A visualization of two consecutive time-triggered steps is shown in Figure 3. Here, $\mathcal{S}$ = POINT, so `sims` is just a single point. Initially, `sims` is α. After δ_{tt} time, the point β is reached; after δ_{tt} further time, the simulation reaches γ. The modification of the output lists after these two steps would be the time-triggered value δ_{tt} twice inserted into list 1, the red rectangle set inserted into list 2, followed by the green rectangle set inserted into list 2.

A **space-triggered step** attempts to use numerical simulations to find a function π for space-triggered splitting, but may, in certain cases, be aborted without modifying the output lists. First, the bounding box of `sims` is computed, bloated by ϵ, and stored in `start`. The center point in `sims`, which we call `p`, is numerically simulated until either, (1) the plane induced by the point lies entirely on one side of `start`, or (2) the space-triggered time limit δ_{pi} is reached. If condition (2) occurs, the space-triggered step returns without modifying the output lists, and reverts the status of `sims`. For condition (1), the plane induced by a point `p` is a hyperplane that both contains `p` and is orthogonal to the gradient at `p`. The function π is created from the equation of the hyperplane, where π is zero along the plane and positive on the side of `start` (in the opposite direction of the gradient at `p`). Forcing transitions along hyperplanes orthogonal to the gradient was previously shown as effective in reducing the size of the currently-tracked set of reachable states in the context of pseudo-invariants [11, 12]. Each of the other points in `sims` are then numerically simulated until either (1) they reach a point along the constructed hyperplane where π evaluates to zero, or (2) they are simulated for the space-triggered time limit δ_{pi}. If for any point condition (2) occurs, again, the space-triggered step aborts without modifying the output lists, and reverts the status of `sims`. If condition (1) occurs for every point in `sims`, the bounding box of all the points in `sims` (which are all along the hyperplane) is taken, bloated by ϵ, and assigned to `end`. The bounding box of `start` and `end` is then computed, and put into output list 2 (contraction domains). The hyperplane function π is put into output list 1.

At the end of the iterative construction, output list 3 is created by performing linearization in each of the contraction domains in list 2, and then solving for the difference between the nonlinear dynamics function and its linearization. This is, in general, a global optimization problem, although guaranteed bounds can be computed using, for example, interval arithmetic. This is also an embarrassingly parallel problem, which can be exploited to speed up this computationally expensive step.

Finally, the last element of list 1 is removed, so that the last mode in the constructed chain will not be split. This process results in three lists, the first of size $N - 1$, and the

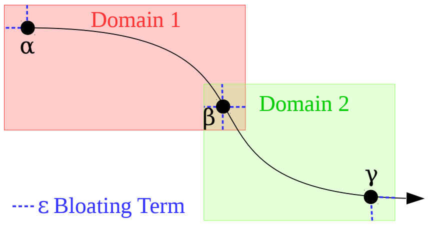


Figure 3: Two time-triggered steps use numerical simulations to create two contraction domains.

other two of size N, as is needed by the proposed mixed-triggered hybridization approach.

5.2 Generalizations

The proposed construction approach is simple in that only a small number of user-parameters are required. However, fine-tuning is possible which can create more precise abstractions, at the cost of requiring more input from the user.

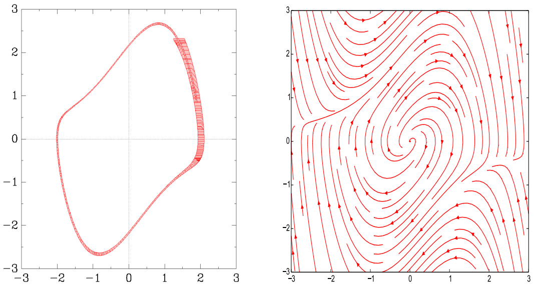
First, the time step δ_{tt} could be changed for each domain. In Figure 3 this would correspond to the case where the difference in simulation times between points α and β is not the same as the difference between β and γ. Next, a per-domain bloating term ϵ is possible. Furthermore, each domain's bloating term could be further parameterized based on the face of the rectangular domain.

The domains need not be rectangles aligned to axes. Domains which are rotated rectangles, aligned with the direction of the flow, could reduce the error in the dynamics abstraction step. As with other hybridization work [21], domains which are triangles (simplices), or rotated variants could also be used. The complication with these approaches is that the global optimization step of domain abstraction, which is necessary for soundness, can become more complicated. For example, the simplex-based approach requires optimizing the Hessian matrix of the dynamics in a simplex domain, which may be difficult depending on the specific location's dynamics.

6. EVALUATION

As stated by Theorem 5.1, in order to soundly reason about the set of reachable states of the original automaton, the output automaton from the mixed-triggered hybridization process must not reach any DCEMs. The main purpose of the evaluation, therefore, is (1) to demonstrate that the hybridization parameters derived from simulations can result in models where DCEMs are not reached during reachability analysis of the output automaton. Additionally, we aim to (2) demonstrate the benefits of occasional space-triggered transitions compared with a pure time-triggered approach. Finally, we (3) demonstrate improved scalability by running our developed static approach on a higher dimensional model, at a granularity that would be impossible for existing static approaches. The evaluation was performed with these three goals in mind.

The proposed hybridization method was implemented in the HYST model translation and transformation tool [12][5]. The developed transformation pass implements the algorithm described in Section 5 leveraging the transformations of Section 4. We target the latest version of the SpaceEx tool, which supports time-triggered transitions using the `map-zero-duration-jump-sets` flag. In order to derive the dynamics abstraction function, we use a global optimization routine from the `scipy.optimize` library. Other options are possible, for example interval arithmetic, interval arithmetic with grid-paving, SMT solvers, or combinations of these methods. Since the optimizations in each domain are run in parallel, more effort can be taken to derive tighter bounds without significant effects on overall runtime. The reported times were measured on a computer with an Intel Core 2 Quad CPU (Q9650) at 3.00 GHz with 4 GB RAM.



(a) Computed reachability (b) Streamplot

Figure 4: The limit cycle for the Van der Pol system was computed with SpaceEx using our hybridization approach.

6.1 Van der Pol Oscillator

The first set of experiments consider a Van der Pol oscillator, which is a two-dimensional system with the following nonlinear dynamics:

$$\dot{x} = y$$
$$\dot{y} = (1 - x^2) * y - x$$

We use the same initial states as evaluated in other hybridization approaches [1], $(x, y) \in [1.25, 1.55] \times [2.28, 2.32]$. A maximum time of 5.5 was used, which is sufficient to complete one cycle of the oscillator, as in the earlier work.

We used numerical simulations based on the $\mathcal{S}$ = STAR strategy, a time-triggered step of $\delta_{tt} = 0.05$, a bloating term of $\epsilon = 0.05$, a number of space-triggered transformation steps of $n_{pi} = 31$, and a maximum simulation time in a space-triggered transformation step of $\delta_{pi} = 1$. Analyzing the generated model with SpaceEx resulted in no DCEMs being reached, which means that the set of reachable states overapproximates the set of reachable states in the the original automaton. This demonstrates goal (1) of the evaluation. The combined hybridization and computation process took 10.3 seconds. A visualization of the resultant set of reachable states produced by SpaceEx is given in Figure 4a, and can be compared to a streamplot of the dynamics given in Figure 4b.

It is insightful to examine the bounding box of the numerical simulations upon entering each mode, and compare it to the bounding box of the set of reachable states at the same times. In particular, by looking at the maximum width in any dimension of the bounding box of `sims` and comparing it with the maximum width of bounding box of the set of reachable states, we can estimate how close the set of reachable states was to the boundaries of the contraction domains where a DCEM would be reached. A plot of these widths upon entering each mode is shown in Figure 5.

[5] SpaceEx model files for the examples evaluated, both before and after hybridization, are available at: http://verivital.com/hyst/pass-hybridization/.

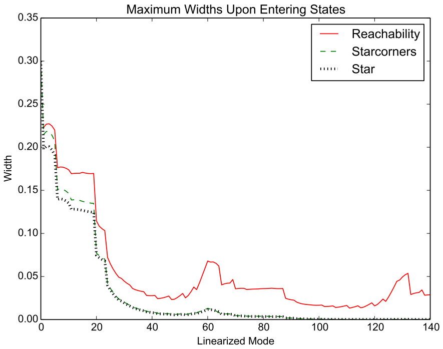


Figure 5: The maximum width of the bounding boxes of the reachable states and simulations upon entering each mode remains within $2 * \epsilon = 0.1$, which is necessary to avoid entering a DCEM.

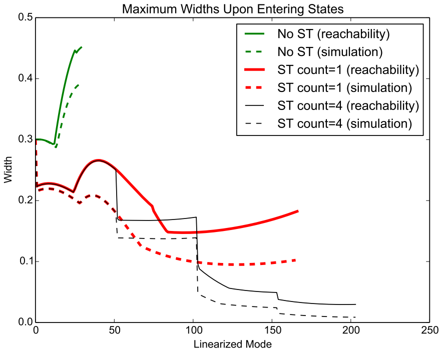
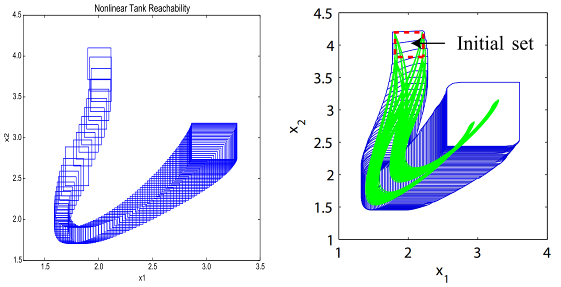


Figure 6: Space-triggered transitions serve to reduce the size of the tracked set of states.

Since we used a bloating term of $\epsilon = 0.05$, it is necessary that maximum width of the simulated states plus $2 * \epsilon = 0.1$ is greater than the maximum width of the set of reachable states at all times, otherwise, an error state will be reached. Additionally, from the plot we can see that the STARCORNERS strategy has slightly better tracking of widths of the set of reachable states near the start of the computation, although it makes less of a difference later on.

In order to show the effect of space-triggered transitions, we consider the same system using a shorter time bound of 2.0, a time step of $\delta_{tt} = 0.01$, and the same value of $\epsilon = 0.05$. We run the system with no space-triggered transitions, a single space-triggered transition at the start, and four space-triggered transitions. The widths of the tracked set of reachable states, and the bounding box of the simulated points is shown in Figure 6. Without space-triggered transitions, the width of the set of reachable states quickly gets larger than the simulated bounding box, and around time 0.29, a DCEM is reached. With a single space-triggered transition at the start, the tracked set of states is smaller, and a DCEM is not reached until around time 1.66. With four space-triggered transitions, the full 2.0 seconds is computed without reaching a DCEM. Furthermore, the decrease in the size of the tracked states is apparent at the space-triggered times 0.0 (mode #0), 0.5 (mode #51), 1.0 (mode #102), and 1.5 (mode #153). This demonstrates the effectiveness of space-triggered transitions in reducing the size of the currently-tracked set of states, goal (2) of the evaluation.



(a) Computed Reachability (b) Result from [5] (includes input disturbances)

Figure 7: A plot of a projection of the computed reachable states for x_1 and x_2 for the 6-D non-linear tank model.

6.2 Nonlinear Water Tank

The next model we consider is a nonlinear tank model [5]. This model is parameterized on the number of tanks, n, where we use $n = 6$. Each tank i adds a single variable x_i to the model, which represents the height of the water in the tank. The input to the first tank is based on the level of the last tank, x_n. We analyze a deterministic version of the model, with no disturbance input and fixed tank parameters. The dynamics for x_1 and every other $x_{i>1}$ are:

$$\dot{x}_1 = 0.1 + 0.01(4 - x_n) + 0.015\sqrt{2gx_1}$$
$$\dot{x}_i = 0.015\sqrt{2gx_{i-1}} - 0.015\sqrt{2gx_i}$$

We used the same initial set of states as the earlier work, $x_1 \in [1.9, 2.1]$, $x_2 \in [3.9, 4.1]$, $x_3 \in [3.9, 4.1]$, $x_4 \in [1.9, 2.1]$, $x_5 \in [9.9, 10.1]$, and $x_6 \in [3.9, 4.1]$. Using the simulation strategy $\mathcal{S}$ = STARCORNERS, a maximum time of $T = 400$, a step size of $\delta_{tt} = 4$, a bloating term value of $\epsilon = 0.2$, a number of space-triggered transformation steps of $n_{pi} = 10$, and a maximum simulation time in a space-triggered transformation step of $\delta_{pi} = 10$, the hybridized model was created. SpaceEx was used to analyze this model, and indicated that no DCEMs were reached. The whole process took about 430 seconds. Figure 7 shows a projection of the set of reachable states onto x_1 and x_2, as well as a result from the earlier hybridization work.

This demonstrates goal (3) of the evaluation, that static-based hybridization approaches can scale to higher dimensions. Although only a six-dimensional model was considered, this is higher than we could find for any published static hybridization method.

7. CONCLUSION

In this paper, we developed the first static time-triggered and mixed-triggered hybridization approaches. The developed methods use simulations to guide the hybridization process and modify an input model for analysis with off-the-shelf verification tools, unlike dynamic hybridization methods that require tool modification. Additionally, we can perform the expensive dynamics abstraction (linearization) step for each mode in parallel, which can improve the speed of the method. We have shown the effectiveness of the method by hybridizing example nonlinear systems and computing the set of reachable states using SpaceEx, a tool that is only capable of reasoning with linear and affine systems.

Since this is the first paper investigating this category of hybridization techniques, we believe significant further optimization is possible. Extending the approach from single-mode input automata to multiple-mode systems would be a straightforward enhancement, and has been done in other hybridization approaches [1]. Dynamic mixed-triggered approaches, have also yet to be investigated. Parameter selection for the approach can also be challenging and could be further automated, perhaps by using a CEGAR-like approach to detect when DCEMs (error modes) are reached, and performing additional simulations from violation regions. Finally, the simulation-based parameter construction algorithm does not track the set of reachable states well when nondeterminism or disturbances are present, and other approaches from hybrid automaton falsification may work better in these cases.

Acknowledgment

The material presented in this paper is based upon work supported by the Air Force Office of Scientific Research (AFOSR), in part under contract number FA9550-15-1-0258 and the Summer Faculty Fellowship Program (SFFP), by AFRL through contract number FA8750-15-1-0105, and by the National Science Foundation (NSF) under grant numbers CNS 1464311 and CCF 1527398. Furthermore, this research was supported in part by the European Research Council (ERC) under grant 267989 (QUAREM) and by the Austrian Science Fund (FWF) under grant numbers S11402-N23 (RiSE) and Z211-N23 (Wittgenstein Award). Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of AFRL, AFOSR, or NSF.

8. REFERENCES

[1] M. Althoff. Reachability analysis of nonlinear systems using conservative polynomialization and non-convex sets. In *Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control*, HSCC '13, pages 173–182, New York, NY, USA, 2013. ACM.

[2] M. Althoff and J. Dolan. Set-based computation of vehicle behaviors for the online verification of autonomous vehicles. In *Intelligent Transportation Systems (ITSC), 2011 14th International IEEE Conference on*, Oct 2011.

[3] M. Althoff and J. Dolan. Online verification of automated road vehicles using reachability analysis. *Robotics, IEEE Transactions on*, 30(4):903–918, Aug 2014.

[4] M. Althoff and B. H. Krogh. Avoiding geometric intersection operations in reachability analysis of hybrid systems. In *Hybrid Systems: Computation and Control, HSCC'12, Beijing, China, April 17-19, 2012*, pages 45–54, 2012.

[5] M. Althoff, O. Stursberg, and M. Buss. Reachability analysis of nonlinear systems with uncertain parameters using conservative linearization. In *47th IEEE Conference on Decision and Control (CDC)*, pages 4042–4048, Dec. 2008.

[6] M. Althoff, O. Stursberg, and M. Buss. Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes. *Nonlinear Analysis: Hybrid Systems*, 4(2), 2010.

[7] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138(1):3–34, 1995.

[8] E. Asarin, T. Dang, and A. Girard. Reachability analysis of nonlinear systems using conservative approximation. In *Hybrid Systems: Computation and Control*, volume 2623 of *LNCS*, pages 20–35. Springer, 2003.

[9] E. Asarin, T. Dang, and A. Girard. Hybridization methods for the analysis of nonlinear systems. *Acta Informatica*, 43:451–476, 2007.

[10] S.-i. Azuma, J.-i. Imura, and T. Sugie. Lebesgue piecewise affine approximation of nonlinear systems. *Nonlinear Analysis: Hybrid Systems*, 4(1):92–102, 2010.

[11] S. Bak. Reducing the wrapping effect in flowpipe construction using pseudo-invariants. In *4th ACM SIGBED International Workshop on Design, Modeling, and Evaluation of Cyber-Physical Systems (CyPhy 2014)*, pages 40–43, 2014.

[12] S. Bak, S. Bogomolov, and T. T. Johnson. HyST: A source transformation and translation tool for hybrid automaton models. In *Proc. of the 18th Intl. Conf. on Hybrid Systems: Computation and Control (HSCC)*. ACM, 2015.

[13] S. Bogomolov, A. Donzé, G. Frehse, R. Grosu, T. T. Johnson, H. Ladan, A. Podelski, and M. Wehrle. Abstraction-based guided search for hybrid systems. In E. Bartocci and C. R. Ramakrishnan, editors, *International SPIN Symposium on Model Checking of Software 2013*, LNCS. Springer, 2013.

[14] S. Bogomolov, A. Donze, G. Frehse, R. Grosu, T. T. Johnson, H. Ladan, A. Podelski, and M. Wehrle. Guided search for hybrid systems based on coarse-grained space abstractions. *Software Tools for Technology Transfer (STTT)*, Aug. 2015.

[15] S. Bogomolov, G. Frehse, M. Greitschus, R. Grosu, C. S. Pasareanu, A. Podelski, and T. Strump. Assume-guarantee abstraction refinement meets hybrid systems. In *10th International Haifa Verification Conference (HVC 2014)*, volume 8855 of *LNCS*, pages 116–131. Springer, 2014.

[16] S. Bogomolov, C. Schilling, E. Bartocci, G. Batt, H. Kong, and R. Grosu. Abstraction-based parameter synthesis for multiaffine systems. In *11th International Haifa Verification Conference (HVC 2015)*, volume 9434 of *LNCS*, pages 19–35. Springer, 2015.

[17] X. Chen, E. Abraham, and S. Sankaranarayanan. Taylor model flowpipe construction for non-linear hybrid systems. *2013 IEEE 34th Real-Time Systems Symposium*, 0:183–192, 2012.

[18] T. Dang. Approximate reachability computation for polynomial systems. In J. Hespanha and A. Tiwari, editors, *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006.

[19] T. Dang, A. Donze, and O. Maler. Verification of analog and mixed-signal circuits using hybrid system techniques. In A. Hu and A. Martin, editors, *Formal Methods in Computer-Aided Design*, volume 3312 of *LNCS*, pages 21–36. Springer, 2004.

[20] T. Dang, C. Le Guernic, and O. Maler. Computing reachable states for nonlinear biological models. In *Computational Methods in Systems Biology*, pages 126–141. Springer, 2009.

[21] T. Dang, O. Maler, and R. Testylier. Accurate hybridization of nonlinear systems. In *Hybrid Systems: Computation and Control (HSCC)*, pages 11–20, New York, NY, 2010. ACM.

[22] P. Duggirala, S. Mitra, M. Viswanathan, and M. Potok. C2e2: A verification tool for stateflow models. In C. Baier and C. Tinelli, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, LNCS, pages 68–82. Springer, 2015.

[23] G. Frehse, R. Kateja, and C. Le Guernic. Flowpipe approximation and clustering in space-time. In *Hybrid Systems: Computation and Control (HSCC'13)*, pages 203–212. ACM, 2013.

[24] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. SpaceEx: Scalable verification of hybrid systems. In *Computer Aided Verification (CAV)*, LNCS. Springer, 2011.

[25] A. Girard. Reachability of uncertain linear systems using zonotopes. In M. Morari and L. Thiele, editors, *Hybrid Systems: Computation and Control*, LNCS. Springer, 2005.

[26] A. Girard and C. Le Guernic. Zonotope/hyperplane intersection for hybrid systems reachability analysis. In M. Egerstedt and B. Mishra, editors, *Hybrid Systems: Computation and Control*, volume 4981 of *LNCS*, pages 215–228. Springer, 2008.

[27] A. Girard, C. Le Guernic, et al. Efficient reachability analysis for linear systems using support functions. In *Proc. of the 17th IFAC World Congress*, pages 8966–8971, 2008.

[28] Z. Han and B. Krogh. Reachability analysis of nonlinear systems using trajectory piecewise linearized models. In *American Control Conference, 2006*, pages 6 pp.–, June 2006.

[29] T. Henzinger, P.-H. Ho, H. Wong-Toi, et al. Algorithmic analysis of nonlinear hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):540–554, 1998.

[30] L. Jaulin, M. Kieffer, and O. Didrit. *Applied interval analysis : with examples in parameter and state estimation, robust control and robotics.* Springer, London, 2001.

[31] T. T. Johnson, J. Green, S. Mitra, R. Dudley, and R. S. Erwin. Satellite rendezvous and conjunction avoidance: Case studies in verification of nonlinear hybrid systems. In D. Giannakopoulou and D. Méry, editors, *Proceedings of the 18th International Conference on Formal Methods (FM 2012)*, pages 252–266. Springer, Paris, France, Aug. 2012.

[32] A. Puri and P. Varaiya. Verification of hybrid systems using abstractions. In *Hybrid Systems II*, LNCS, pages 359–369. Springer, 1994.

Software Tools for Technology Transfer manuscript No.
(will be inserted by the editor)

Hybrid automata: from verification to implementation

**Stanley Bak[1], Omar Ali Beg[2], Sergiy Bogomolov[3],
Taylor T. Johnson[4], Luan Viet Nguyen[2], Christian Schilling[5]**

[1] Air Force Research Laboratory, USA,
[2] University of Texas at Arlington, USA
[3] Australian National University, Australia and IST Austria, Austria
[4] Vanderbilt University, USA
[5] University of Freiburg, Germany

The date of receipt and acceptance will be inserted by the editor

Abstract Hybrid automata are an important formalism for modeling dynamical systems exhibiting mixed discrete-continuous behavior such as control systems and are amenable to formal verification. However, hybrid automata lack expressiveness compared to integrated model-based design (MBD) frameworks such as the MathWorks' Simulink/Stateflow (SLSF). In this paper, we propose a technique for correct-by-construction compositional design of cyber-physical systems (CPS) by embedding hybrid automata into SLSF models. Hybrid automata are first verified using verification tools such as SpaceEx, and then automatically translated to embed the hybrid automata into SLSF models such that the properties verified are transferred and maintained in the translated SLSF model. The resultant SLSF model can then be used for automatic code generation and deployment to hardware, resulting in an implementation. The approach is implemented in a software tool building on the HyST model transformation tool for hybrid systems. We show the effectiveness of our approach on a CPS case study—a closed-loop buck converter—and validate the overall correct-by-construction methodology: from formal verification to implementation in hardware controlling an actual physical plant.

1 Introduction

In this paper, we present the theory and associated implementation for the translation of hybrid automaton models (used for verification) to the MathWorks Simulink/Stateflow (SLSF) models, subsequently used for design refinement, simulation, implementation, and code generation for target embedded hardware. Our approach is particularly useful if the design process is structured in a bottom-up fashion. In other words, we assume that the individual system components are first modeled in detail, such as modeling a control algorithm as a hybrid automaton and verifying properties (typically safety) for it. These components are then linked together to form the whole system under consideration within SLSF. This leads to overall system models consisting of heterogeneous components where a number of components are modeled as hybrid automata, but the entire system may be too complex to formally model and verify. In the last decade, a number of powerful formal design, analysis, and verification tools for hybrid automata such as SpaceEx [9–12, 22] and Flow* [17] have emerged. In our proposed approach, a designer can ensure the correctness of individual components before using our translation process to link the system together in SLSF (see Fig. 1).

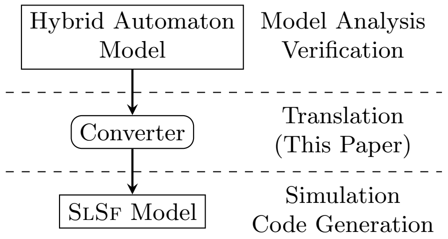


Figure 1: High-level overview of the model-based design process enabled by this work. Verification using the hybrid automaton is first performed in a hybrid systems model checker, then we automatically generate a trajectory-equivalent SLSF diagram. The diagram can then be embedded into a more complex system, possibly with other, unverified, components (because they are too large to verify, exist for legacy reasons, etc.), and can then be used for code generation and implementation in actual systems.

DISTRIBUTION A. Approved for public release; Distribution unlimited. (Approval AFRL PA #88ABW-2015-2402)

We introduce a technique to automatically convert the hybrid automata into *trajectory-equivalent* SLSF diagrams. By trajectory-equivalent, we mean that behaviors (trajectories) of the translated SLSF diagram match those of the original hybrid automaton. One technical challenge is that hybrid automata and SLSF differ in semantics: a hybrid automaton is typically defined with *may*-semantics with respect to the discrete transitions, whereas SLSF employs *must*-semantics. In other words, a transition in SLSF is taken as soon as the transition guard is enabled subject to some numerical aspects with zero-crossing detection, whereas the hybrid automaton still has the freedom to stay in the current location as long as the location invariant has not been violated. In case of non-deterministic hybrid automata, trajectory-equivalence means that the behaviors of the original hybrid automaton will be exhaustively explored. Our approach incorporates additional randomization steps into the resulting SLSF diagram. In this way, in every run, the diagram produces a possibly different trace that still reflects a trajectory from the original hybrid automaton semantics. After running more and more simulations, we get a better and better approximation of the reachable state space of the original hybrid automaton.

Related Work Significant research has been done on the translation of SLSF diagrams into other analysis tools, such as hybrid systems model checkers [2, 4, 8, 14, 15, 29–31, 36, 37, 40, 42]. Agrawal et al. [2] suggest an algorithm to translate SLSF diagrams into the equivalent HSIF [14, 15, 36, 37] models. The Compositional Interchange Format (CIF) provides a common input language focused on model compositionality for networks of hybrid automata [3]. Alur et al. translated SLSF to linear hybrid automata for applying symbolic analysis to improve test coverage of SLSF [4]. In a different setting, Schrammel et al. [40] consider the translation problem for complex SLSF diagrams where involved treatment of zero-crossings is needed. Manamcheri et al. [29] have developed the tool HyLink to translate a restricted class of SLSF to hybrid automata. Minopoli et al. [30, 31] have developed a theory of urgent semantics for hybrid automata and the SL2SX tool that translates a restricted subset of SLSF diagrams to hybrid automata. The application of the above techniques is restricted by the fact that no complete semantics of SLSF is provided (in spite of recent progress [8, 13, 23, 24, 29, 38]).

In contrast to all these existing works, we consider the converse direction, i.e., to translate a given hybrid automaton into an SLSF diagram. Sanfelice et al. [39] have developed the hybrid equations toolbox (HyEQ) to approximately simulate the hybrid systems that may include Zeno, zero-crossing, and non-deterministic behaviors. However, the applicability of the Simulink Design Verifier (SDV) model checker[1] integrated with SLSF does not apply to this class of models, so verification is not possible. In our setting, we benefit from clear and unambiguous hybrid automata semantics and may formally verify properties of the hybrid automata prior to translating them to SLSF diagrams. Pajic et al. [25, 33–35] consider a similar problem of converting timed automata encoded in UPPAAL [27] to SLSF diagrams. However, in their translation, they consider only runs of UPPAAL models that obey the *must*-semantics. In our work, beyond considering the much more expressive framework of hybrid automata (as timed automata are a subclass of hybrid automata), we provide a translation handling the non-determinism by producing trajectory-equivalent SLSF diagrams. Operational semantics of (purely discrete) Stateflow have been developed [24], and alternative formalizations of discrete semantics have been investigated using, e.g., translation from Stateflow to C [38]. In contrast to these prior works, we focus on continuous-time Stateflow diagrams. Another recent line of research focusses on the translation from Hybrid Communicating Sequential Processes (HCSP) to Simulink block diagrams [16, 43, 44]. In our work we consider the translation of the hybrid automaton model which is extensively used in the industry for CPS modeling.

Contributions. This paper has four primary contributions.

(a) This is the first work, as far as we are aware, to provide a translation scheme from hybrid automata to SLSF diagrams, which is useful as part of a model-based design (MBD) process. *(b)* In order to overcome the difference in semantics between the modeling frameworks, we introduce the notion of trajectory-equivalence, and show how the conversion preserves trajectory-equivalence with respect to several sources of non-determinism in hybrid automata. *(c)* We provide an implementation of the trajectory-equivalent translation scheme as a part of the HyST model translation framework [6], which enables completely automatic translation of existing hybrid automaton models. *(d)* We show the applicability of our contributions in several case studies where hybrid automata are automatically translated to SLSF for simulation, use in larger SLSF diagrams, and deployment to actual hardware. For one case study—a closed-loop buck converter—the entire correct-by-construction MBD process is illustrated, from verification through implementation in hardware. This includes formal verification of the hybrid automaton in SpaceEx, translation to SLSF, code generation for the controller in SLSF, then subsequent compilation, and finally execution in embedded hardware controlling the physical plant.

Paper Organization. The remainder of the paper is organized as follows. After introducing the necessary background in Sect. 2, we present our trajectory-equivalent translation scheme in Sect. 3. In Sect. 4, we evaluate our approach on four case studies. We conclude in Sect. 5.

[1] `http://www.mathworks.com/products/sldesignverifier/`

2 Preliminaries

In this section, we introduce the preliminaries that are needed for this work. We first define a hybrid automaton model and discuss its semantics, and then do the same for SLSF diagrams.

2.1 Hybrid Automata

A hybrid automaton is formally defined as follows.

Definition 1 (Hybrid Automaton). A *hybrid automaton* is a tuple $\mathcal{H} \triangleq (Loc, Var, Init, Flow, Trans, Inv)$ with: *(a)* the finite set of locations *Loc*, *(b)* the set of continuous variables $Var \triangleq \{x_1, \ldots, x_n\}$ from $\mathbb{R}^n$, *(c)* the initial condition, given by $Init(\ell) \subseteq \mathbb{R}^n$ for each location ℓ, *(d)* the flow, a deterministic function $Flow(\ell)$ from the variables to their derivatives for each location ℓ, *(e)* the discrete transition relation *Trans*, where every transition is a tuple $(\ell, g, \upsilon, \ell')$ with: *(i)* the source location ℓ and the target location ℓ', *(ii)* the guard, given by a constraint g, *(iii)* the update, given by a mapping υ that modifies the variable valuation, and *(f)* the invariant $Inv(\ell) \subseteq \mathbb{R}^n$ for each location ℓ.

We use the common . (dot) notation to specifically indicate components of $\mathcal{H}$ as necessary, e.g., $\mathcal{H}.Var$ are the variables of $\mathcal{H}$.

The semantics of a hybrid automaton $\mathcal{H}$ is defined in terms of trajectories as follows. A *state* of $\mathcal{H}$ is a pair $(\ell, \mathbf{x})$ that consists of a location $\ell \in Loc$ and a point $\mathbf{x} \in \mathbb{R}^n$. Formally, $\mathbf{x}$ is a valuation of the continuous variables in *Var*. For the following definitions, let $T = [0, \Delta]$ be an interval for some $\Delta \geq 0$.

Definition 2. A *trajectory* of $\mathcal{H}$ from state $s = (\ell, \mathbf{x})$ to state $s' = (\ell', \mathbf{x}')$ is a pair $\rho \triangleq (L, \mathbf{X})$, where $L : T \to Loc$ and $\mathbf{X} : T \to \mathbb{R}^n$ are functions that define for each time point in T the location and the values of the continuous variables, respectively. A sequence of time points where location switches happen in ρ is denoted by $(\tau_i)_{i=0\ldots k} \in T^{k+1}$. In this case, we define the *length* of ρ as $|\tau| = k$. Trajectories $\rho = (L, \mathbf{X})$, and the corresponding sequence $(\tau_i)_{i=0\ldots k}$, must satisfy the following conditions:

(a) $\tau_0 = 0$, $\tau_i < \tau_{i+1}$, and $\tau_k = \Delta$ – the sequence of switching points increases, starts with 0 and ends with Δ,

(b) $L(0) = \ell$, $\mathbf{X}(0) = \mathbf{x}$, $L(\Delta) = \ell'$, $\mathbf{X}(\Delta) = \mathbf{x}'$ – the trajectory starts in $s = (\ell, \mathbf{x})$ and ends in $s' = (\ell', \mathbf{x}')$,

(c) $\forall\, i\ \forall\, t \in [\tau_i, \tau_{i+1}) : L(t) = L(\tau_i)$ – the location is not changed during the continuous evolution,

(d) $\forall\, i\ \forall\, t \in [\tau_i, \tau_{i+1}) : \ (\mathbf{X}(t), \dot{\mathbf{X}}(t)) \in Flow(L(\tau_i))$ holds and thus the continuous evolution is consistent with the differential equations of the corresponding location,

(e) $\forall\, i\ \forall\, t \in [\tau_i, \tau_{i+1}) : \ \mathbf{X}(t) \in Inv(L(\tau_i))$ – the continuous evolution is consistent with the corresponding invariants, and

(f) $\forall\, i < k\ \exists\, (L(\tau_i), g, \upsilon, L(\tau_{i+1})) \in Trans : \mathbf{X}_{end}(i) \in g \wedge \mathbf{X}(\tau_{i+1}) = \upsilon(\mathbf{X}_{end}(i)) \wedge \mathbf{X}_{end}(i) = \lim_{\tau \to \tau_{i+1}^-} \mathbf{X}(\tau)$ – every continuous transition is followed by a discrete one, where $\mathbf{X}_{end}(i)$ defines the values of continuous variables immediately before the discrete transition at the time moment τ_{i+1}.

A state s' is *reachable* from state s if there exists a trajectory from s to s'.

A *symbolic state* $s \triangleq (\ell, \mathcal{R})$ is a pair, where $\ell \in Loc$ and $\mathcal{R}$ is a convex and bounded set consisting of points $\mathbf{x} \in \mathbb{R}^n$. The continuous part $\mathcal{R}$ of a symbolic state is also called *region*. The symbolic state space of $\mathcal{H}$ is called the *region space*. The initial set of states $\mathcal{S}_{init}$ of $\mathcal{H}$ is defined as $\bigcup_\ell (\ell, Init(\ell))$. The reachable state space $\mathrm{Reach}(\mathcal{H})$ of $\mathcal{H}$ is defined as the set of symbolic states that are reachable from some initial state in $\mathcal{S}_{init}$, where the definition of reachability is extended accordingly for symbolic states. We refer to the set of all the trajectories of $\mathcal{H}$ starting in $\mathcal{S}_{init}$ by $\mathrm{Traj}(\mathcal{H})$. A *safety specification* P is a given set of symbolic states. A hybrid automaton $\mathcal{H}$ *satisfies* a safety specification P iff $\mathrm{Reach}(\mathcal{H}) \subseteq P$. We are interested in ensuring that the hybrid automaton is correct, i.e., satisfies P, and then subsequently translate it for simulation, integration, and implementation in SLSF as discussed in the next sections.

2.2 Continuous-Time Stateflow Diagrams

Simulink is a graphical modeling language for control systems, plants, and software. Stateflow is a state-based graphical modeling language integrated within Simulink. Continuous-time Stateflow diagrams provide methods for modeling hybrid systems that consist of continuous and discrete states and behaviors. In this section, we describe a *restricted subclass of continuous-time Stateflow diagrams* to which we translate a hybrid automaton. In particular, we focus only on continuous-time Stateflow state transition diagrams and we do not consider models with hierarchical states.

Roughly, a Stateflow state transition diagram may be thought of as an extended state machine with variables of various types. In addition to states, Stateflow diagrams may have junctions that are instantaneous. A transition between states may occur at each simulation time step, whereas multiple junction transitions may occur in a single simulation time step.

A continuous-time Stateflow diagram (see Fig. 2) is roughly analogous to a hybrid automaton, but their behavior differs in several ways. In particular, Stateflow diagrams (1) are deterministic, (2) have urgent transitions with priorities, and (3) have events such as enabled transitions that are determined at runtime by zero-crossing detection algorithms.

We define Stateflow diagrams more formally now.

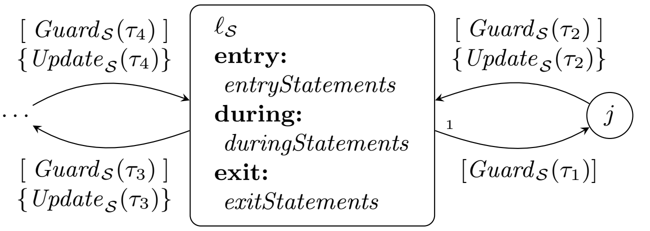

[$Guard_{\mathcal{S}}(\tau_4)$]
{ $Update_{\mathcal{S}}(\tau_4)$ }
...
[$Guard_{\mathcal{S}}(\tau_3)$]
{ $Update_{\mathcal{S}}(\tau_3)$ }
$\ell_{\mathcal{S}}$
entry:
entryStatements
during:
duringStatements
exit:
exitStatements
[$Guard_{\mathcal{S}}(\tau_2)$]
{ $Update_{\mathcal{S}}(\tau_2)$ }
1
j
[$Guard_{\mathcal{S}}(\tau_1)$]

Figure 2: Snippet of a general continuous-time Stateflow diagram with a state $\ell_{\mathcal{S}}$, a junction j, and four transitions $\tau_1 - \tau_4$.

Definition 3 (Stateflow diagram). The tuple $\mathcal{S} \triangleq (Loc_{\mathcal{S}}, Junc_{\mathcal{S}}, Var_{\mathcal{S}}, Trans_{\mathcal{S}}, Actions_{\mathcal{S}})$ defines the *Stateflow diagram*. Here, *(a)* $Loc_{\mathcal{S}}$ is a finite set of *states* (also known as *locations*), *(b)* the junctions $Junc_{\mathcal{S}}$ are like locations, but all of which may be evaluated in a single simulation event step (i.e., they are instantaneous "states"), *(c)* $Var_{\mathcal{S}}$ is a finite set of variables of various types, and for our formalization we assume they are real-valued, *(d)* the $Actions_{\mathcal{S}}(\ell_{\mathcal{S}})$ for each location $\ell_{\mathcal{S}}$ are actions described by Matlab or C statements that are performed at different event times subdivided into `entry`, `during`, and `exit` actions, where the `entry` (resp. `exit`) action is executed only once when entering (resp. exiting) the state and the `during` action performs the continuous-time evolution of the variables of $Var_{\mathcal{S}}$ according to a differential equation (this happens strictly between entering and exiting), *(e)* the discrete transition relation $Trans_{\mathcal{S}}$ where every transition $\tau \in Trans_{\mathcal{S}}$ is formally defined as a tuple $(\ell_{\mathcal{S}}, Guard_{\mathcal{S}}, Update_{\mathcal{S}}, TP_{\mathcal{S}}, \ell'_{\mathcal{S}})$: *(i)* the source location or junction $\ell_{\mathcal{S}} \in Loc_{\mathcal{S}} \cup Junc_{\mathcal{S}}$ and the target location or junction $\ell'_{\mathcal{S}} \in Loc_{\mathcal{S}} \cup Junc_{\mathcal{S}}$, *(ii)* the guard, given by a constraint $Guard_{\mathcal{S}}$, must be satisfied for a transition to be taken, *(iii)* the update, given by a mapping $Update_{\mathcal{S}}$, defines which variables in $Var_{\mathcal{S}}$ are modified, and to what value (unmodified variables keep their value), and *(iv)* the priority, given by $TP_{\mathcal{S}}$, is a natural number between 1 and $od(\ell_{\mathcal{S}})$—the outdegree of (number of transitions leaving) the state or junction $\ell_{\mathcal{S}}$—that indicates the order in which transitions are taken if more than one is enabled.

Simulating an SLSF diagram produces a simulation trajectory, which is closely related to a trajectory of a hybrid automaton.

Definition 4 (Simulation trajectory). For an initial state x_0, a time bound $\mathcal{T}_{\max}$, error bound $\delta \geq 0$, and time step $\tau > 0$, a *simulation trajectory* (of length k) is a sequence $\alpha \triangleq ((R_i, t_i))_{i=1...k}$, where $R_0 = \{x_0\}$, $t_0 = 0$, $R_i \subseteq \mathbb{R}^n$, $t_i \in \mathbb{R}^{\geq 0}$, and *(a)* $\forall i : 0 \leq t_{i+1} - t_i \leq \tau$, $t_k = \mathcal{T}_{\max}$, *(b)* $\forall i \; \forall t \in [t_i, t_{i+1}]$: the simulation state after time t is in R_i, and *(c)* $\forall i : dia(R_i) \leq \delta$.

Here $dia(\cdot)$ denotes the diameter and δ is used to bloat the simulation trajectory to handle numerical errors; picking $\delta = 0$ represents the typical result of a (idealized) numerical simulation of an SLSF diagram. We note that the various actions (e.g., `entry`, `during`, and `exit` actions, and transition updates) are evaluated sequentially, while hybrid automaton actions are executed concurrently. By $\mathrm{Trac}_\delta(\mathcal{S})$ we denote the set of all simulation trajectories of an SLSF diagram $\mathcal{S}$ with parameter δ. A simulation trajectory α *satisfies* a safety specification P if every element $\alpha.R_i \subseteq P$, i.e., P contains the states of the simulation trajectory with time projected away. An SLSF diagram $\mathcal{S}$ *satisfies* a safety specification P if all simulation trajectories $\mathrm{Trac}_\delta(\mathcal{S})$ satisfy P. Note that in practice, any simulation trajectory is finite-length, although we avoid a finite-length assumption in the definition of simulation trajectories to relate possibly infinite trajectories of a hybrid automaton with similar possibly infinite simulation trajectories. Moreover note that our definition of a trajectory does not allow instantaneous location switches in the hybrid automaton. This restriction is necessary for practical purposes because SLSF requires executing a (however small) simulation step in each state.

3 Translating a Hybrid Automaton to a Continuous-Time Stateflow Diagram

We describe our main contribution, namely how to translate from a hybrid automaton to an SLSF diagram. For different classes of hybrid automata, different translations may be used, and we discuss two classes primarily based on whether the hybrid automaton is deterministic or not.

To compare simulation trajectories of an SLSF diagram with trajectories of a hybrid automaton, we introduce the concept of correspondence. Here we assume that the δ parameter of a simulation trajectory is equal to zero.

Definition 5 (Correspondence). A trajectory ρ of a hybrid automaton $\mathcal{H}$ and a simulation trajectory α (with $\delta = 0$) of an SLSF diagram $\mathcal{S}$ *correspond* to each other if the sequences of discrete locations, transitions, and transition times encountered in both are the same, and the continuous points of the trajectory and the simulation trajectory match.

The primary goal of our construction is to ensure that the set of simulation trajectories $\mathrm{Trac}_\delta(\mathcal{S})$ for the SLSF diagram can be *trajectory-equivalent* to the original hybrid automaton.

Definition 6 (Trajectory-Equivalence). An SLSF diagram $\mathcal{S}$ is *trajectory-equivalent* to a hybrid automaton $\mathcal{H}$ if, for every trajectory ρ of $\mathcal{H}$, there exists a corresponding (Definition 5) simulation trajectory α of $\mathcal{S}$, and for every simulation trajectory α of $\mathcal{S}$, there exists a corresponding trajectory ρ of $\mathcal{H}$.

3.1 Translating different classes of hybrid automata

As already outlined in Sect. 1, one main difference between hybrid automata and SLSF diagrams is the absence of *non-determinism* in SLSF diagrams. There are several sources of non-determinism in the general hybrid automaton formalism.

1. *Transitions.* If there is more than one outgoing transition in a location, any of them can be taken as long as the guard is enabled and the target location's invariant is satisfied after applying the transition update.
2. *Dwell times.* The amount of time that a hybrid automaton remains in a location is only determined by the invariant and the transition guards – it is forced to leave the location *only* by the invariant. It is not sufficient for the guard to be enabled at *some* point in time, as the automaton can still choose to remain in the location until the invariant becomes false.
3. *Initial states.* A hybrid automaton is allowed to start in a whole region, which may be an uncountable number of possible initial states.
4. *Updates.* Updates in transitions may be non-deterministic. This gives a (possibly uncountable) number of successor states after a discrete transition.
5. *Flows.* Flow definitions in locations may be uncertain. We do not consider this source of non-determinism in this paper.

For the translations, we make the following assumptions on the original hybrid automaton.

Assumption 1 *The hybrid automaton $\mathcal{H}$ is Zeno-free, which means that only finitely many discrete transitions may be taken in finite time.*

Translating deterministic hybrid automata is fairly straightforward, so we first discuss how to translate deterministic hybrid automata, and then discuss the more complex non-deterministic scenario. There may be additional numerical issues with SLSF that are outside the scope of this work. For example, the integration of the differential equations in SLSF may not be exact, which may cause differences in observed behavior. In practice, simulations can be made arbitrarily accurate by reducing the simulation time step at a computational cost.

3.1.1 Translating a deterministic hybrid automaton

The next definition states when a hybrid automaton is deterministic.

Definition 7. A hybrid automaton $\mathcal{H}$ is *deterministic* if, for any initial state $(\ell, x_0) \in \mathcal{S}_{init}$ for any point $x_0 \in Init(\ell)$, there is one unique trajectory ρ starting from (ℓ, x_0). Otherwise, $\mathcal{H}$ is *non-deterministic.*

Syntactic restrictions may be enforced on a hybrid automaton to ensure it is deterministic. For example, a sufficient condition for a hybrid automaton to be deterministic includes all of the following being satisfied: (1) at most one discrete transition is enabled simultaneously, (2) a discrete transition guard is enabled when the continuous flow exits the invariant, and (3) no state can be mapped onto two different states by the transition updates [26, Lemma 2]. Note that requirement (2) is not an urgent definition of semantics, but it is a condition that ensures an enabled transition is forced to occur once it becomes enabled, so it is in essence a syntactic restriction that enforces urgency.

Under such assumptions that enforce a hybrid automaton to be deterministic, the translation from the deterministic hybrid automaton to an SLSF diagram is straightforward and proceeds as follows. Let $\mathcal{S} = (Loc_{\mathcal{S}}, Junc_{\mathcal{S}}, Var_{\mathcal{S}}, Trans_{\mathcal{S}}, Actions_{\mathcal{S}})$ be the SLSF diagram. Instantiate $Loc_{\mathcal{S}} = \mathcal{H}.Loc$, $Junc_{\mathcal{S}} = \emptyset$, and $Var_{\mathcal{S}} = \mathcal{H}.Var$. For each location $\ell \in Loc$ and each corresponding location $\ell_{\mathcal{S}} \in Loc_{\mathcal{S}}$, and for each variable $v \in Var$ and the corresponding variable $v_{\mathcal{S}} \in Var_{\mathcal{S}}$, we set the $Actions_{\mathcal{S}}(\ell_{\mathcal{S}}, v_{\mathcal{S}})$ `during` action for $v_{\mathcal{S}}$ to be equal to the flow $Flow(\ell, v)$ for variable v, and do not instantiate the `entry` and `exit` actions. For continuous-time Stateflow models, the `during` action is used to specify an ordinary differential equation for variables, so in essence this just copies the flow from $\mathcal{H}$ to $\mathcal{S}$ for each location and each variable, and the other action types (`entry` and `exit`) are unused.

Finally, we instantiate the transitions as follows. For each location $\ell \in Loc$ and corresponding location $\ell_{\mathcal{S}} \in Loc_{\mathcal{S}}$, and for each transition $(\ell, g, v, \ell') \in Trans$ with a natural number i indicating the iteration count over the transitions, we instantiate a transition $\gamma \in Trans_{\mathcal{S}}$ as the tuple $(\ell_{\mathcal{S}}, Guard_{\mathcal{S}}, Update_{\mathcal{S}}, TP_{\mathcal{S}}, \ell'_{\mathcal{S}})$, where $\gamma.\ell_{\mathcal{S}} = \ell$, $\gamma.Guard_{\mathcal{S}} = g$, $\gamma.Update_{\mathcal{S}} = v$, $TP_{\mathcal{S}} = i$, and $\gamma.\ell'_{\mathcal{S}} = \ell'$. Since $\mathcal{H}$ is deterministic, the choice of the transition priority $TP_{\mathcal{S}}$ is unimportant as only at most one transition is enabled at a time, so it is in essence set arbitrarily to i based on whatever iteration order is chosen. Additionally, the restriction on guards and invariants to ensure determinism means the invariant translation is naturally handled through the translation of the guard as described above.

There are some additional minor syntactic translations that also must occur which we discuss briefly. The first is due to the fact that updates in SLSF are evaluated sequentially, whereas in a hybrid automaton they are evaluated concurrently, so additional temporary variables are introduced to handle this as necessary (e.g., the hybrid automaton update $x' := x + 1 \wedge y' := x$ is rewritten to the SLSF update $x'_{tmp} := x; x' := x_{tmp} + 1; y' := x_{tmp}$, where x_{tmp} is a fresh temporary variable).

The second more significant difference is related to how SLSF identifies events during execution or simulation, which is influenced in part by the simulator not be infinitely precise and have numerical errors. In particular, this influences event detection such as when transitions are enabled and may be taken, and this is implemented using zero-crossing detection algorithms inside the simulation routines of SLSF.

In particular, if a guard is only enabled at one (singular) point in time, it will almost surely not be detected by the zero-crossing mechanisms used by SLSF, and the transition is usually missed. In order to not exclude certain behaviors systematically, we consider an ε-relaxation of each guard constraint, similar to the relaxations considered in translations from SLSF to hybrid automata [30]. For instance, a guard constraint of the form $x = c \wedge y \leq x$ becomes $c - \varepsilon \leq x \leq c + \varepsilon \wedge y \leq x - \varepsilon$. The simulation time step can then be chosen small enough such that, based on the value of ε and the Lipschitz constant of the dynamics, no transitions will be missed.

Although this may permit more behaviors than the original hybrid automaton, it critically prevents transitions from being missed, which is necessary for trajectory-equivalence. The extra behaviors introduced from this necessary step can be reduced by considering smaller values of ε, which will require a smaller simulation time step. Reducing the time step, however, will be at the cost of additional simulation runtime.

Example Translation. We illustrate the translation process with a running case study evaluated in more detail later (Section 4.1). A deterministic hybrid automaton for this example appears in Figure 3, which is a model of a closed-loop control system. Specifically, here a periodically-updated hysteresis controller is used to regulate a voltage V_C by controlling the state of a switch. This is a flattened (composed) model of the closed-loop system, originally consisting of a timed automaton model of the hysteresis controller which has periodic updates every 20 microseconds, and a hybrid automaton model with affine dynamics of the plant, which is a circuit known as a buck converter. The resulting continuous-time Stateflow diagram for the buck converter created using our translator appears in Figure 4 (with no ε-relaxations).

3.1.2 Translating a non-deterministic hybrid automaton

For a non-deterministic hybrid automaton, we achieve trajectory-equivalence by replacing non-determinism in the hybrid automaton by (uniformly distributed) random number generation in the SLSF diagram. In this way, by executing multiple SLSF simulations we can approximate the reachable states of the original hybrid automaton.

In our converter, we currently support initial regions and non-deterministic updates to hyper-rectangles, as well as deterministic updates which can be arbitrary functions. When non-deterministic assignments or initial states are used, they must be strict subsets of the invariant of the target or initial location, respectively, which we note can be statically checked. Under this assumption, the choice of the initial continuous state and the non-determinism possible during updates can be done by randomly choosing one point from the set of all points available.

In the rest of this section, we focus on the harder problem of non-determinism from the transitions and dwell time. We first give an overview of the translation scheme. Here it is helpful to regard the trajectory of a hybrid automaton as a sequence of jumps, and after each jump, the automaton chooses the next transition and dwell time. The crucial difference in our conversion is that the choices might be infeasible, i.e., violating the invariant. To account for this, we incorporate a backtracking mechanism, where the current state of all variables is stored when entering a new location. Note that *time* is an entity which is implicitly present in all hybrid automaton models and we can always add a (fresh) time variable t with flow $\dot{t} = 1$. This allows for a general translation scheme without further knowledge about the hybrid automaton under consideration.

We translate a hybrid automaton location ℓ into a corresponding *location cluster* $\hat{\ell}$, comprising of a number of SLSF states, junctions, and transitions. The clusters are then connected by the same transitions as in the original hybrid automaton. A simulation trajectory of the resulting SLSF diagram then visits those clusters. Inside a cluster, the execution consists of three *phases*, depicted in Fig. 5.

Three phases in a location cluster. In the first phase, we *randomly* choose a transition *out* from the transitions currently available. In the second phase, we choose a time threshold $\mathcal{T}$. In the final phase, we incorporate the original continuous dynamics of the location ℓ.

In the translated model, the transition tries to be taken by checking the original guard condition, but only after dwelling in $\hat{\ell}$ for at least until time moment $\mathcal{T}$. If the transition *out* cannot be taken – possibly due to an invariant violation – in the time frame $[\mathcal{T}, \mathcal{T}_{\max}]$, where $\mathcal{T}_{\max}$ is the maximum simulation time, we *backtrack*[2] and return to the second phase, and select a new time threshold $\mathcal{T}$ which is strictly less than the previously-chosen threshold. To ensure termination, we bound the number of times backtracking may occur before trying $\mathcal{T} = 0$. If the chosen transition can still not be taken, we can conclude that it cannot be taken at all, and go back to the first phase, this time trying another transition.

3.2 Trajectory-Equivalence

The translation process described above maintains the defined notion of trajectory-equivalence. For this, we consider an idealized conversion, where there are no numerical errors in the simulation, the value of ε is zero,

[2] We note that our notion of backtracking is different from the one that occurs with multiple junctions in SLSF. In particular, we require allowing some dwell time to elapse in states, whereas junctions are instantaneous.

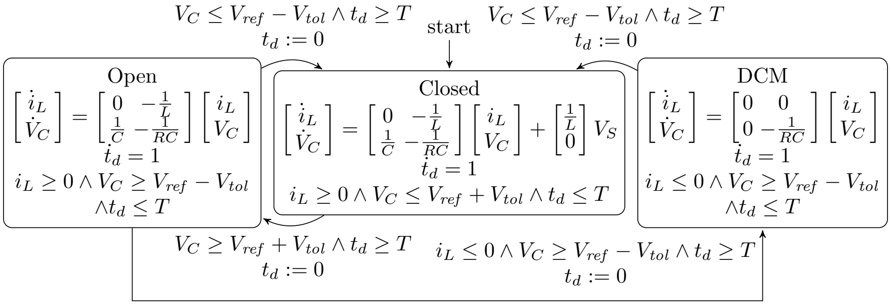

$V_C \leq V_{ref} - V_{tol} \wedge t_d \geq T$
$t_d := 0$
start
$V_C \leq V_{ref} - V_{tol} \wedge t_d \geq T$
$t_d := 0$
Open
$\begin{bmatrix}\dot{i}_L \\ \dot{V}_C\end{bmatrix} = \begin{bmatrix}0 & -\frac{1}{L} \\ \frac{1}{C} & -\frac{1}{RC}\end{bmatrix}\begin{bmatrix}i_L \\ V_C\end{bmatrix}$
$\dot{t}_d = 1$
$i_L \geq 0 \wedge V_C \geq V_{ref} - V_{tol} \wedge t_d \leq T$
Closed
$\begin{bmatrix}\dot{i}_L \\ \dot{V}_C\end{bmatrix} = \begin{bmatrix}0 & -\frac{1}{L} \\ \frac{1}{C} & -\frac{1}{RC}\end{bmatrix}\begin{bmatrix}i_L \\ V_C\end{bmatrix} + \begin{bmatrix}\frac{1}{L} \\ 0\end{bmatrix} V_S$
$\dot{t}_d = 1$
$i_L \geq 0 \wedge V_C \leq V_{ref} + V_{tol} \wedge t_d \leq T$
DCM
$\begin{bmatrix}\dot{i}_L \\ \dot{V}_C\end{bmatrix} = \begin{bmatrix}0 & 0 \\ 0 & -\frac{1}{RC}\end{bmatrix}\begin{bmatrix}i_L \\ V_C\end{bmatrix}$
$\dot{t}_d = 1$
$i_L \leq 0 \wedge V_C \geq V_{ref} - V_{tol} \wedge t_d \leq T$
$V_C \geq V_{ref} + V_{tol} \wedge t_d \geq T$
$t_d := 0$
$i_L \leq 0 \wedge V_C \geq V_{ref} - V_{tol} \wedge t_d \geq T$
$t_d := 0$

Figure 3: Composed hybrid automaton model of the closed-loop feedback control system for the buck converter. The buck converter plant is originally modeled as a hybrid automaton and the hysteresis controller is modeled as a timed automaton(see Figure 11).

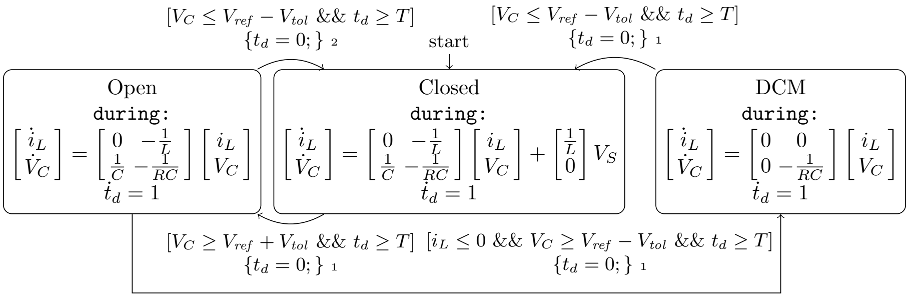

$[V_C \leq V_{ref} - V_{tol}$ && $t_d \geq T]$
$\{t_d = 0;\}$ 2
start
$[V_C \leq V_{ref} - V_{tol}$ && $t_d \geq T]$
$\{t_d = 0;\}$ 1
Open
during:
$\begin{bmatrix}\dot{i}_L \\ \dot{V}_C\end{bmatrix} = \begin{bmatrix}0 & -\frac{1}{L} \\ \frac{1}{C} & -\frac{1}{RC}\end{bmatrix}\begin{bmatrix}i_L \\ V_C\end{bmatrix}$
$\dot{t}_d = 1$
Closed
during:
$\begin{bmatrix}\dot{i}_L \\ \dot{V}_C\end{bmatrix} = \begin{bmatrix}0 & -\frac{1}{L} \\ \frac{1}{C} & -\frac{1}{RC}\end{bmatrix}\begin{bmatrix}i_L \\ V_C\end{bmatrix} + \begin{bmatrix}\frac{1}{L} \\ 0\end{bmatrix} V_S$
$\dot{t}_d = 1$
DCM
during:
$\begin{bmatrix}\dot{i}_L \\ \dot{V}_C\end{bmatrix} = \begin{bmatrix}0 & 0 \\ 0 & -\frac{1}{RC}\end{bmatrix}\begin{bmatrix}i_L \\ V_C\end{bmatrix}$
$\dot{t}_d = 1$
$[V_C \geq V_{ref} + V_{tol}$ && $t_d \geq T]$
$\{t_d = 0;\}$ 1
$[i_L \leq 0$ && $V_C \geq V_{ref} - V_{tol}$ && $t_d \geq T]$
$\{t_d = 0;\}$ 1

Figure 4: Composed SLSF diagram for the translated closed-loop feedback control system for the buck converter.

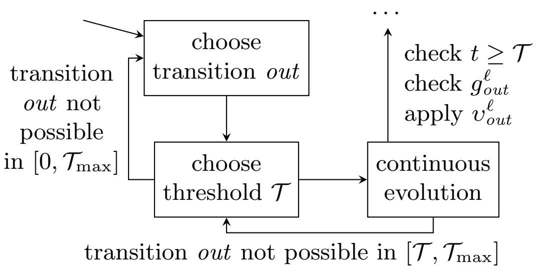

choose transition *out*
transition *out* not possible in $[0, \mathcal{T}_{\max}]$
choose threshold $\mathcal{T}$
continuous evolution
...
check $t \geq \mathcal{T}$
check g^{ℓ}_{out}
apply v^{ℓ}_{out}
transition *out* not possible in $[\mathcal{T}, \mathcal{T}_{\max}]$

Figure 5: High-level location cluster translation pattern consisting of three phases. The location cluster $\hat{\ell}$ denotes a group of SLSF states and junctions which reflects the behavior of the hybrid automaton in the location ℓ.

and the SLSF diagram encodes the intended semantics of the described transformation process.

Theorem 1. *If $\mathcal{H}$ is a Zeno-free hybrid automaton and $\mathcal{S}$ is the* SLSF *diagram created using our transformation process, then $\mathcal{S}$ is trajectory-equivalent to $\mathcal{H}$.*

The proof for the more complex non-deterministic case is given in the Section 3.3.4. From the theorem we can conclude that our translation preserves safety properties.

Corollary 1. *If a Zeno-free hybrid automaton $\mathcal{H}$ satisfies a safety specification P, then every simulation trajectory of the translated* SLSF *diagram $\mathcal{S}$ also satisfies P.*

3.3 Additional Translation Details and Proof

3.3.1 Detailed Translator Description

We provide a detailed description of our translation. It iteratively converts every location ℓ of a hybrid automaton and its outgoing transitions into an SLSF diagram of location clusters $\hat{\ell}$ in the following way (see Fig. 6). We first describe the data structures we use in our construction. The list *outList* stores the ordering in which the outgoing transitions of the location ℓ are considered in the simulation. The variable *out* keeps track of the currently chosen outgoing transition. The variable $\mathcal{T}_v$ stores the first time moment when the location invariant is violated. $\mathcal{T}_{\max}$ keeps the maximum simulation time, i.e., the simulation is stopped as soon as this bound has been reached. The variable $\mathcal{T}$ stores the time threshold after which the outgoing transition should be taken. The variable R keeps the maximum number of backtrackings we want to allow, whereas r stores the current number of backtrackings in the location cluster $\hat{\ell}$. Finally, the variable t stores the current time that is simulated. Introducing this variable allows us to model going back in time when backtracking, which is not possible for the actual simulation time that is tracked by SLSF.

We continue with the description of every individual (SlSf) state in our construction. The current simulation time and the hybrid automaton state when entering the location ℓ (and respectively the location cluster $\hat{\ell}$) is stored in the (SlSf) state ℓ_{in}. Furthermore, the algorithm *randomly* chooses the ordering in which the outgoing transitions are considered. In this way, we handle the non-determinism due to multiple simultaneously enabled transition guards. Finally, the variable $\mathcal{T}_v$ is initialized to $\mathcal{T}_{\max}$ as we do not have any information about the invariant violation at that moment.

The state ℓ_{choose} covers two kinds of non-determinism. It takes care of the situation when the intersection of the invariant and the transition guard is non-singular, i.e., when a switch to the next location can happen not only at a particular time moment, but within a *time interval*. Note that if the continuous dynamics are non-monotonic, there can be multiple *disjoint* time intervals where the guard is enabled. We resolve such situations by generating a *random* time threshold $\mathcal{T}$ in the state ℓ_{choose} and allowing the discrete transition only from the time moment $\mathcal{T}$ onward, i.e., we add a constraint of the form $t \geq \mathcal{T}$ as a part of the transition guard for every outgoing transition from the location ℓ. Thus, we disable the SlSf must-semantics up until time moment $\mathcal{T}$ to mimic the original may-semantics of hybrid automata.

Note that we also use the state ℓ_{choose} for *backtracking* purposes. We observe that an unfortunate choice of the outgoing transition *out* and the time threshold $\mathcal{T}$ can lead to the simulation getting stuck, as the transition guard of *out* is not enabled in the time frame $[\mathcal{T}, \mathcal{T}_{\max}]$ and thus the transition cannot be taken. In such cases, we return to the state ℓ_{choose} to select a further time threshold $\mathcal{T}$. For this purpose, we restore the simulation time t and the state of the hybrid automaton from the moment we entered ℓ resp. $\hat{\ell}$. Afterward, we can choose the next time threshold from the interval $[t, \mathcal{T}]$. Here we observe that in general before reaching the time threshold, the invariant can be violated. Thus, we actually select a new threshold from the interval $[t, \min(\mathcal{T}, \mathcal{T}_v)]$. In this way, we end up with a sequence of monotonically decreasing thresholds. Still, as it is not guaranteed that the chosen threshold is eventually equal to 0, we add a further termination criterion by bounding the number of backtracking by some user-defined constant $R > 0$. The last time before exceeding this limit, we try out the weakest threshold $\mathcal{T} = 0$ to ensure that we have covered all cases. If the transition cannot be taken at all, we either proceed with a further outgoing transition (junction j_{in}) or, if none is left, the simulation is stopped and reports an actual deadlock in the model.

The continuous evolution corresponding to the location ℓ is modeled by the state ℓ_{dwell}. We can leave this state under two conditions. First, the invariant can be violated. Then we store the time moment when the violation has happened in the variable $\mathcal{T}_v$ and move to the state ℓ_{choose} (via junction j_{v}). Note that if we have already considered all the outgoing transitions of ℓ, we will stop the simulation since a deadlock has been found. In the other case, the time threshold $\mathcal{T}$ can be reached. We take the transition to the successor location of ℓ if the guard of the chosen transition *out* is enabled and after applying the update, the target location's invariant is satisfied (junction j_{t}). Furthermore, here we also check whether the maximum simulation time $\mathcal{T}_{\max}$ has been reached, in which case we stop the simulation.

In the following, we illustrate the translation process using an example simulation.

3.3.2 Example

We consider an execution in some location cluster for a simple location ℓ_1 with one continuous variable x and two outgoing transitions, as depicted in Fig. 7. For simplicity, assume that the location is entered at time $t = 0$ in state $x = 0$ and the total simulation time is $\mathcal{T}_{\max} = 20$.

First we store the current continuous state $(t, x) = (0, 0)$. Next, in phase 1, we choose a transition, say, the one to ℓ_2. Then, in phase 2, we choose a random minimum dwell time in the range $[0, 20]$, say $\mathcal{T} = 3$. The simulation proceeds in phase 3 until an event occurs. In this case, events are either violating the location invariant $x < 10$ or enabling the guard condition of the selected transition $t \geq 3 \wedge x \geq 8$. The guard condition is enabled first, at state $(t, x) = (4, 8)$. This transition cannot be taken, however, as the target invariant would be violated after applying the update $x := 2$.The simulation continues until the next event, when the state $(t, x) = (5, 10)$ is reached and a violation of the invariant is detected. That is why the simulation goes back to phase 2, backtracking to the saved state $(t, x) = (0, 0)$. At this point, it was checked that for all $\mathcal{T} \geq 3$, the transition cannot be taken. In phase 2, a new value for $\mathcal{T}$ is chosen from the restricted interval $[0, 3)$, and the simulation is run again in phase 3. After reaching the same conclusion and after further backtracking, a finite threshold of attempts is reached, and $\mathcal{T} = 0$ is forced. Even with $\mathcal{T} = 0$ there will be a violation of the invariant before the transition can be taken. Then, we will conclude that the selected transition can never be taken when starting in the state $(t, x) = (0, 0)$. Thus we can safely ignore this transition, go back to phase 1 and choose the transition leading to ℓ_3, where the process repeats.

3.3.3 Translation Correctness and Discussion

Correctness. The proof of Theorem 1 required three assumptions, mentioned before the theorem statement and proven below. First, we assumed the simulations were exactly accurate. Although real simulations will always have some error, this can be reduced to arbitrarily small values by reducing the time step used in the simulation. Similarly, for the second assumption we can con-

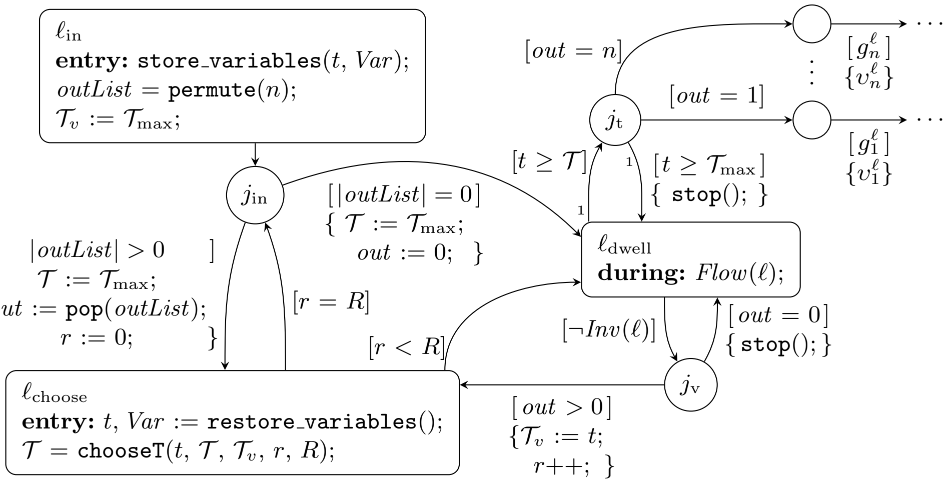


Figure 6: General location cluster of some location ℓ with n outgoing transitions. (re-)store_variables stores and restores the current simulation state (including the time variable t) from when entering the cluster, respectively. permute(n) returns a permuted list $outList$ with all integers from 1 to n. pop($outList$) removes and returns the first element from $outList$. chooseT chooses a new time threshold $\mathcal{T}$. A subscript "1" indicates that a transition has the highest priority among all the outgoing transitions from a state/junction.

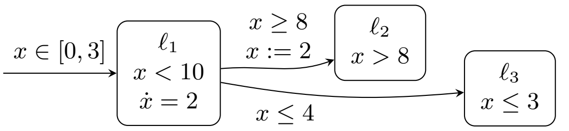


Figure 7: Snippet of an example hybrid automaton with three locations $\ell_1 - \ell_3$.

sider smaller and smaller values of ε, although in degenerate cases this might permit extra transitions in the simulation. For example, a degenerate guard like $x < 5 \wedge x > 5$ will always be false, but any positive ε-relaxation will have a possible transition when $5 - \varepsilon < x < 5 + \varepsilon$. The third assumption is that the SLSF diagram correctly encodes the described transformation process. This means that correctness is subject to possible implementation bugs in our conversion implementation in HyST, as well as the semantics of Stateflow. In addition to the trajectory-equivalence theorem, we provide empirical justification for the correctness of the implementation of our translation scheme, through extensive case studies including the buck converter detailed in the main body, and additional case studies presented later in the appendix.

Non-determinism. By replacing non-determinism with random number generation, some behaviors of the original hybrid automaton might be obscured. For instance, a non-deterministic die can roll a six forever, while the probability of this behavior for a random die approaches zero as more rolls are taken. We always deal with finite executions in a simulation, and thus end up with a finite number of choices, so there is still a nonzero chance that the 'right' random values will be chosen, assuming that the hybrid automaton is Zeno-free.

Generalizations. Although we consider a large class of hybrid automata, further generalizations are possible. For example, the initial sets and non-deterministic resets in our framework were hyper-rectangles, whereas in general the initial state could be in a non-convex set, and the reset might be an arbitrary function which maps from a single state to a non-convex set. To handle such systems, we need a way to sample in the non-convex destination sets, which may be possible in certain situations, but is difficult in general. One possibility would be to require the user to give this sampling function.

Another generalization possible is to consider non-deterministic dynamics. More general hybrid automata may include differential inclusions or other non-deterministic ways for the continuous states to evolve. This could be handled by adding ranged inputs to the system, and at each time step choosing a random value in the range for each input. However, as the time steps become smaller, the random inputs will approximate the main value in their ranges, which in practice results in poor simulation coverage. An alternative is to choose a time step where the inputs will vary, such that a tradeoff is possible between the amount of coverage possible, and the effect of this tendency towards the mean. Other simulation methods, perhaps based on state exploration mechanisms such as rapidly-exploring random trees (RRTs) [28] may also be possible.

3.3.4 Proof

Proof (Theorem 1). We first show the forward direction, i.e., given an arbitrary trajectory of the hybrid automaton, there exists a set of random decisions in the constructed SLSF diagram that produce a corresponding simulation trajectory.

Recall that correspondence (Definition 5) requires that the encountered locations can be the same, and that the deviation in continuous states can be bounded by an arbitrarily small constant.

For the ordering of locations, notice that the random choice of an outgoing transition in phase 1 of the construction can pick the corresponding transition from the trajectory. Since the minimum dwell time is chosen randomly, it can be picked to be arbitrarily close to the dwell time in the hybrid automaton trajectory. In this way, as long as the continuous evolution in the simulation remains close to the hybrid automaton trajectory's continuous evolution, every transition will be explored.

The second part of correspondence requires that the deviation in the continuous states is bounded. We show that this bound can be chosen to be arbitrarily small across both every continuous evolution and after every discrete transition. During a continuous evolution, if the start state in a location in the simulation is chosen close to the start state in the corresponding location in the hybrid automaton trajectory, its deviation will also be bounded as a function of the Lipschitz constant (see Proposition 1 in [20]). Thus, for a single bounded continuous evolution and every nonzero final state deviation desired, there is a corresponding nonzero initial state deviation that will achieve the desired closeness.

During initial state selection, since we consider hyperrectangles, the set of states is bounded. Randomly choosing states, we will in finite time pick one arbitrarily close to any trajectory's start state in the hybrid automaton.

Finally, for updates, the dwell time of a simulation can be made arbitrarily close to a hybrid automaton trajectory, and since the state can be made arbitrarily close, a deterministic update function (under assumptions of Lipschitz continuity) can also result in a state arbitrarily close to the trajectory. For nondeterministic updates, the argument is similar to the initial state selection, and thus the continuous states of the simulation remain arbitrarily close to the hybrid automaton trajectory.

The sequence of discrete transitions between the trajectory and simulation match. Since each trajectory is a finite sequence of discrete transitions (due to Zeno-free behavior) and continuous evolutions (each of which can have arbitrarily small error between the trajectory and a possible simulation), the accumulated error for the whole trajectory can also be made arbitrarily small. Thus, the constructed SLSF diagram has simulations which correspond to any arbitrary hybrid automaton trajectory.

The reverse direction in the proof shows that any arbitrary simulation has a corresponding hybrid automaton trajectory. Again, we proceed by decomposing this into showing that the sequence of locations is the same, and that the deviation in the continuous state is bounded.

Since we assumed an idealized relaxation where ε is zero, every transition in the simulation exactly matches the guard conditions in the hybrid automaton, and thus the hybrid automaton can match the simulation. Every update in the constructed SLSF diagram is also copied from the automaton, so that the automaton's trajectory can match the random choices made by a simulation.

For continuous trajectories, the simulation will choose some dwell time where the invariant remains satisfied until a guard becomes true. The hybrid automaton can also pick the same dwell time, and its invariant will also remain true until the same guard condition is reached. Thus, the hybrid automaton can pick a trajectory which corresponds to the simulation.

Since every trajectory of the hybrid automaton corresponds to a simulation trajectory of the SLSF diagram, and every simulation trajectory corresponds to a trajectory, the two models are trajectory-equivalent. □

4 Evaluation and Experimental Results

To evaluate the translation methodology presented in this paper, we implemented a prototype translator that uses the HyST intermediate representation for source-to-source transformation of hybrid automata [6], and the SLSF API within Matlab (tested with versions 2014a through 2016a). The input to the translator is a hybrid automaton $\mathcal{H}$ in the SpaceEx XML format. Networks of hybrid automata are first composed within HyST to yield a single hybrid automaton representing the network. Once parsed in the tool, an object representing the syntactic structure of $\mathcal{H}$ is traversed, and then the tool applies the sequence of translation steps described in Sect. 3. In the simulator, we varied the seeds of the uniform pseudo-random number generator `rng` in Matlab. We evaluated the prototype tool using several examples. For this we first computed the reachable states of the models in SpaceEx or Flow*, then performed the translation and simulations in SLSF. The tool and examples are available for download [1].

4.1 Case Study: Buck Converter with Periodic Hysteresis Controller

A buck converter is a DC-to-DC switched-mode power supply that takes a DC input source voltage and lowers ("bucks") it to a smaller DC output voltage [32]. A standard model of the converter has three modes, where: the switch is closed and the voltage source is connected, the switch is open and the voltage source is disconnected, and based on the possible dynamics of the converter, a third mode, known as the discontinuous conduction mode (DCM), where the current is not allowed to go below zero (which is physically unrealizable, but may occur without this third mode). Interested readers may find detailed derivations of models in power electronics textbooks [41]. A hybrid automaton model of the closed-loop buck converter (plant and timed controller) appears in Fig. 3.

A standard closed-loop controller for the buck converter is a hysteresis controller, which changes the mode of the buck converter plant based on the measured output voltage. Its operation depends on opening and closing the MOSFET switch. Intuitively, it operates like a thermostat, i.e., the switch is toggled so that the source voltage is connected to the circuit if the output voltage is too low, and it is toggled in case if the output voltage is too high to disconnect the voltage source. We note that by Kirchhoff's voltage law (KVL), $V_C = V_{out}$ [41]. In part to avoid switching too frequently, a hysteresis band is typically used so switches occur when $V_{out} \geq V_{ref} + V_{tol}$ or $V_{out} \leq V_{ref} - V_{tol}$. This creates a voltage ripple on the output voltage that should be within a given range V_{rip} of the desired reference output voltage V_{ref}. Together, these define a safety specification: $P(t) \triangleq t \geq t_s \Rightarrow V_{out}(t) = V_{ref} \pm V_{rip}$, which projected onto the phase space is $P \triangleq V_{ref} - V_{rip} \leq V_{out} \leq V_{ref} + V_{rip}$. SpaceEx is used to verify P by computing the reachable states Reach($\mathcal{H}$) (to a fixed-point) from a startup state where the initial states $\mathcal{S}_{init}$ are $i_L = 0$ and $V_C = 0$. For every time $t \geq t_s$ after a startup trajectory of duration t_s, if $V_{ref} - V_{rip} \leq V_{out}(t) \leq V_{ref} + V_{rip}$, then the converter satisfies the specification P.

For actual implementations, the measured voltage values are sensed periodically through an analog-to-digital converter (ADC), and subsequently, the control signals are sent periodically to control the state of the buck converter transistor (open/closed). We model this periodic update process as a timed automaton for the controller with a timer variable t_d that evolves at unit rate and is upper bounded by T of 20 microseconds. The reachable states of the closed-loop buck converter hybrid automaton are computed with SpaceEx, and as shown in Fig. 8, the model satisfies the safety specification P for a sufficient choice of V_{rip}.

A hardware setup consisting of a buck converter plant and a dSpace DS1103 is used to perform the experiments with the physical buck converter plant. The DS1103 contains a Power PC processor and a DSP board and is used for implementation of the hybrid automata in both hardware-in-the-loop (HiL) simulations with a "virtual plant" (the plant model simulated on the DS1103 hardware) and the actual buck converter plant.

The hysteresis controller is executed on the DS1103. First, we generate C code using the translated SLSF diagram in Matlab, then compile it and download it onto the DS1103. A discrete fixed-step solver with a time step of 20 microseconds is used for the code generation process and also for the DS1103's sampling and control periods, which is sufficiently small to ensure ε is sufficiently small, as discussed in Sect. 3. The measured voltage signal from the buck converter is periodically sensed and sent to the embedded controller through an ADC. The embedded controller generates Boolean valued signals and these are converted to suitably spaced rectangular pulses to operate the MOSFET switch of the buck converter plant. For the experiments with the actual plant, the input signals fed to the controller (specifically the V_C voltage) are replaced from the simulation model with the measurement of the actual plant, and the output signals (the desired mode, open or closed) are fed to the actual plant instead of the simulation model. The experimental results are recorded and a comparison to SLSF simulations is shown in Fig. 8. The experimental and simulation traces are contained in the SpaceEx reach sets, which validates the translation correctness (Theorem 1) and that the safety property is maintained in the implementation (Corollary 1). Note that in the hardware experiments, the controller has essentially been determinized, as the purpose of non-determinism in the hybrid automaton model was to model plant inaccuracies.

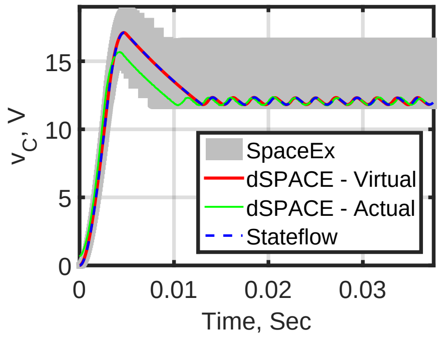


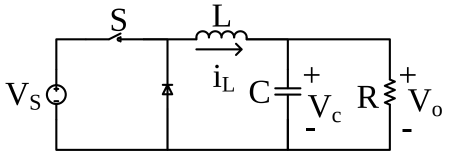
Figure 8: Reachable states of the hybrid automaton computed with SpaceEx, verifying the voltage-regulation property, along with HiL simulation results of the translated SLSF diagram on the DS1103 ("virtual plant"), and control of the physical plant with the translated SLSF diagram ("actual plant"). Our results validate the high-level vision of correct-by-construction control implementation from Fig. 1.

4.1.1 Additional Details

The buck converter circuit appears in Fig. 9(a). Parameter values used for the case study appear in Figure 9(b).

A hybrid automata network model of the buck converter plant and a timed automaton of the hysteresis controller appears in Fig. 11, where θ is a synchronization label and δ is a discrete control signal, and a bisimilar hybrid automaton model after flattening (composing) the network was shown earlier in Fig. 3. The composed model from Fig. 3 is used for verification, translation, and code generation purposes as discussed earlier, while the network model is conceptually simpler and illustrates the decomposition between the physical plant hardware and the controller. The physical hardware used in the evaluation appears in Fig. 10.

Fig. 13 shows the reachable states in the phase space, and illustrates that the SLSF simulations are contained in the reachable states computed with SpaceEx and gives empirical evidence for the correctness of the translation.



(a)

Component / Parameter Name	Symbol	Value
Source Input Voltage	V_S	24 V
Desired Output (Reference) Voltage	V_{ref}	12 V
Actual Output Voltage	$V_C = V_{out}$	12 V ± V_{rip}
Hysteresis Band Tolerance	V_{tol}	0.1 V
Voltage Ripple Tolerance	V_{rip}	0.6 V
Load Resistance with Parameter Variation	R	$10 \pm 2\%$ Ω
Capacitor Value with Parameter Variation	C	$2.2 \pm 2\%$ mF
Inductor Value with Parameter Variation	L	$2.65 \pm 2\%$ mH
Periodic Updation Parameter	T	20 μ sec

(b)

Figure 9: (a) Buck converter circuit—a DC input V_S is decreased to a lower DC output $V_C = V_o = V_{out}$. (b) Buck converter parameter values and variations.

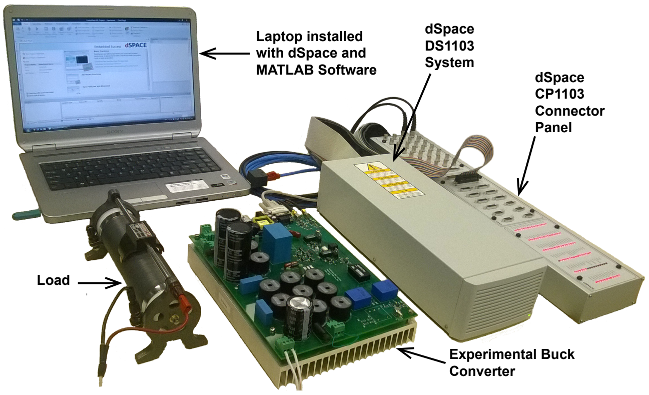

Laptop installed with dSpace and MATLAB Software
dSpace DS1103 System
dSpace CP1103 Connector Panel
Load
Experimental Buck Converter

Figure 10: The buck converter plant controlled with a dSPACE DS1103 system. Our results controlling the actual plant with the translated controller validate the high-level vision of correct-by-construction control implementation from Fig. 1.

4.2 Case Study: Yaw Damper Controller for 747 Aircraft

A yaw damper is modeled as a multiple-input multiple-output (MIMO) system which uses the aileron and rudder in order to reduce oscillations in the yaw and roll angle of an aircraft. In this section, we use the proposed method to analyze the control design of a yaw damper for a 747 aircraft, taken from the Control Systems Toolbox case studies in Matlab.

In particular, we analyze the final designed controller, which includes a washout filter capable of eliminating oscillations, but maintaining the spiral mode. The spiral mode is a desired control characteristic in yaw damper systems, where an impulse input from the aileron will result in a bank angle which does not immediately decrease to zero.

The model for the system is given at Mach 0.8 at 40,000 ft using standard linear time-invariant dynamics, $\dot{x} = Ax + Bu$. There are four physical variables in the system $x = (x_1, x_2, x_3, x_4)^T$, which are sideslip angle (x_1), yaw rate (x_2), roll rate (x_3), and bank angle (x_4), represented by the column vector x. The two inputs $u = (u_1, u_2)^T$, are the rudder (u_1) and aileron (u_2). The outputs are the yaw rate and bank angle.

The specific values for A and B are:

$$A = \begin{bmatrix} -0.0558 & -.9968 & 0.0802 & 0.0415 \\ 0.598 & -0.115 & -0.0318 & 0 \\ -3.05 & 0.388 & -0.4650 & 0 \\ 0 & 0.0805 & 1 & 0 \end{bmatrix}, B = \begin{bmatrix} .00729 & 0 \\ -0.475 & 0.00775 \\ 0.153 & 0.143 \\ 0 & 0 \end{bmatrix}$$

This physical system is put into a feedback loop with a washout filter, which has a single variable w and dynamics $\dot{w} = x_2 - 0.2 \cdot w$. The filter variable is combined with the yaw to produce an effect on the rudder input. In particular, the washout filter adds to u_1 the value $2.34 \cdot (x_2 - 0.2 \cdot w)$.

We consider analysis of a system model which has the guarantees given by a real-time scheduler, which periodically executes the washout filter and sets the output values. Between controller executions we take the output of the washout filter to be constant (zero-order hold). The control task is guaranteed to execute every period using a common scheduler like Rate Monotonic (RM) or Earliest Deadline First (EDF). There is non-determinism in the exact time the controller runs, however, due to the offset of the execution of the control task within each period. Since the control logic is simple, we take the control task to be nonpreemptive and short, so that the model will sample the physical system and update the filter output at a single point in time, but that point in time may vary within each period. Furthermore, we look at the system response due to an impulse input from the aileron from a range of start conditions. We take the initial bank angle to be between 0 and 0.1.

This system was modeled in SpaceEx, and reachability analysis was attempted in both SpaceEx and Flow*.

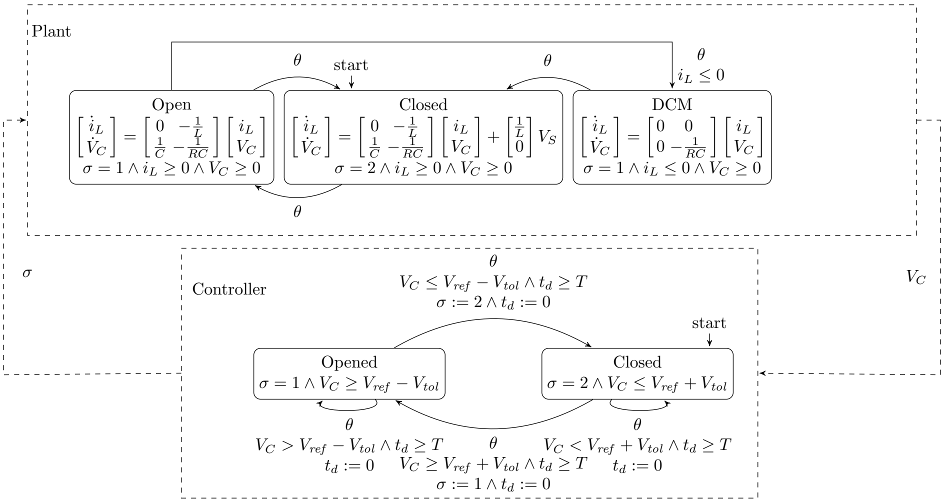

Plant
Open
$\begin{bmatrix}\dot{i}_L\\ \dot{V}_C\end{bmatrix} = \begin{bmatrix}0 & -\frac{1}{L}\\ \frac{1}{C} & -\frac{1}{RC}\end{bmatrix}\begin{bmatrix}i_L\\ V_C\end{bmatrix}$
$\sigma = 1 \wedge i_L \geq 0 \wedge V_C \geq 0$
Closed
$\begin{bmatrix}\dot{i}_L\\ \dot{V}_C\end{bmatrix} = \begin{bmatrix}0 & -\frac{1}{L}\\ \frac{1}{C} & -\frac{1}{RC}\end{bmatrix}\begin{bmatrix}i_L\\ V_C\end{bmatrix} + \begin{bmatrix}\frac{1}{L}\\ 0\end{bmatrix}V_S$
$\sigma = 2 \wedge i_L \geq 0 \wedge V_C \geq 0$
DCM
$\begin{bmatrix}\dot{i}_L\\ \dot{V}_C\end{bmatrix} = \begin{bmatrix}0 & 0\\ 0 & -\frac{1}{RC}\end{bmatrix}\begin{bmatrix}i_L\\ V_C\end{bmatrix}$
$\sigma = 1 \wedge i_L \leq 0 \wedge V_C \geq 0$
θ
start
θ
θ
$i_L \leq 0$
θ
σ
V_C
Controller
θ
$V_C \leq V_{ref} - V_{tol} \wedge t_d \geq T$
$\sigma := 2 \wedge t_d := 0$
start
Opened
$\sigma = 1 \wedge V_C \geq V_{ref} - V_{tol}$
Closed
$\sigma = 2 \wedge V_C \leq V_{ref} + V_{tol}$
θ
$V_C > V_{ref} - V_{tol} \wedge t_d \geq T$
$t_d := 0$
θ
$V_C \geq V_{ref} + V_{tol} \wedge t_d \geq T$
$\sigma := 1 \wedge t_d := 0$
θ
$V_C < V_{ref} + V_{tol} \wedge t_d \geq T$
$t_d := 0$

Figure 11: Hybrid automaton model of the buck converter plant with timed automaton of the hysteresis controller as a network.

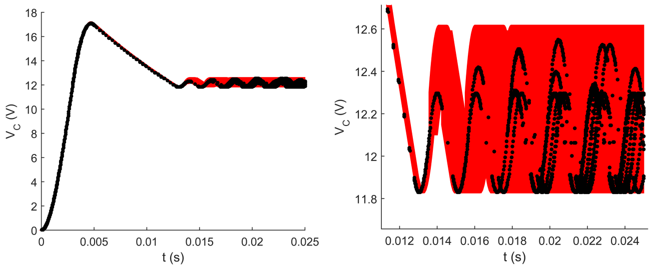


Figure 12: Left: Buck converter V_C versus time, with SpaceEx reach set for the hybrid automatom model in red, and black points from 10 simulation traces of the translated SLSF diagram. Right: Detailed and zoomed view illustrating multiple simulation trajectories.

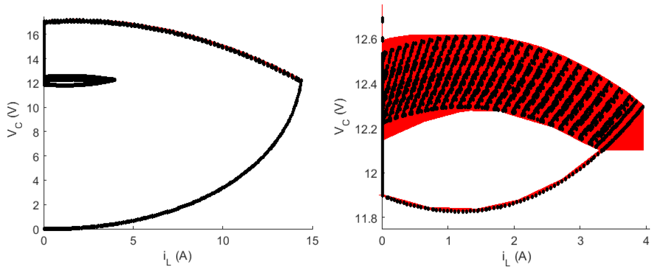


Figure 13: Left: Buck converter V_C versus i_L (phase space), with SpaceEx reach set in red, and black points from 100 simulation traces. Right: Detailed and zoomed view illustrating multiple simulation trajectories.

Due to the large number of discrete switches, however, neither tool is able to directly compute reachability (the computed reach sets grow exponentially).

Instead, we investigate the system using our conversion to SLSF and randomized execution. Since the main source of non-determinism in this model is the discrete switches, we can investigate simulations of the system where they occur at varying offsets from the start of each period.

The simulations showed the expected response of the system when using a controller period of $T = 0.1$. The response of the system is shown in Fig. 14. Here, the impulse response from the aileron to the bank angle is plotted, which does not immediately converge (spiral mode), and does not contain excessive oscillations. Thus, using the technique proposed in this paper we are able to analyze a system which cannot be directly analyzed using reachability tools.

This system can be analyzed formally, however this requires a non-trivial model transformation using the technique of continuization, as well as using a smaller control period. Continuization converts the periodically-actuated model into a continuous one with bounded noise, where the bound is based on the controller period and maximum rate of change of the output signal [7]. The same model can be used as the basis for the conversion using continuization, as well as the conversion to SLSF for simulation and further Matlab-based analysis and code generation. In this way, the conversion to SLSF is one part of a larger toolflow, where models are first created in SpaceEx, possibly converted for formal analysis using HyST, and then can be directly imported into SLSF after the conversion described in this paper for simulation and controller synthesis, as well as embedding in a larger CPS model.

4.3 Case Study: Glycemic Control in Diabetics

Glycemic control is an approach to control the blood glucose levels in insulin dependent diabetes mellitus patients. There are several different mathematical models of glycemic control used to design insulin infusion devices that help diabetic patients control their blood glucose levels [21]. Here we investigate a nonlinear hybrid

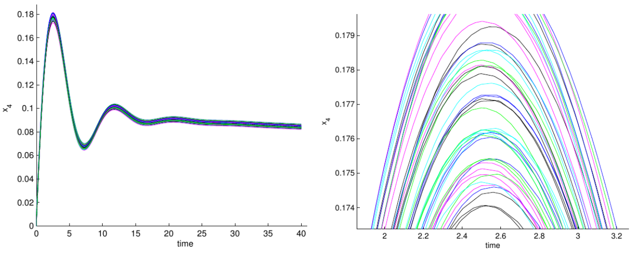


Figure 14: 50 simulations of the yaw damper system. Left: The spiral mode is confirmed. Right: Nondeterminism in controller execution time causes simulated trajectories to cross.

system of the glycemic control in diabetic patients such that all dynamics are defined by polynomials. The mathematical model is described by the following ODEs:

$$\dot{G} = -0.01G - X(G + G_B) + g(t) \tag{1}$$

$$\dot{X} = -0.025X + 0.000013I \tag{2}$$

$$\dot{I} = -0.093(I + I_B) + u(t)/12 \tag{3}$$

In Equation 1 and Equation 3, G and I are the plasma glucose concentration and the plasma insulin concentration above their basal value G_B and I_B, which are equal to 4.5 and 15, respectively. The variable X shown in Equation 2 is the insulin concentration in an interstitial chamber. Moreover, $g(t)$ and $u(t)$ are the influx of glucose and the insulin control input, presented in Equation 4 and Equation 5, respectively.

$$g(t) = \begin{cases} t/60 & \text{if } t \leq 30 \\ (120 - t)/180 & \text{if } 30 < t \leq 120 \\ 0 & \text{if } t > 120 \end{cases} \tag{4}$$

$$u(t) = \begin{cases} 25/3 & \text{if } G(t) \leq 4 \\ 25/3(G(t) - 3) & \text{if } 4 < G(t) \leq 8 \\ 125/3 & \text{if } G(t) > 8 \end{cases} \tag{5}$$

The glycemic control was first modeled in SpaceEx and then translated to Flow* by using the HyST model converter. This model is nonlinear, non-deterministic, and includes 4 variables, 9 locations and 18 discrete transitions in total. The simulations of the glycemic control model translated to SLSF are shown in Fig. 15. We simulated the translated model with 100 different randomized executions. All simulation traces of G are contained in the reach set computed by Flow*, which validates the translation.

4.4 Case Study: Fischer Mutual Exclusion

Fischer mutual exclusion is a timed distributed algorithm that ensures a mutual exclusion safety property,

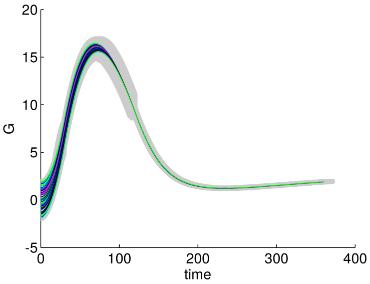


Figure 15: 100 simulations of the glycemic control model with simulations and reach set computed by Flow* (gray) for variable G.

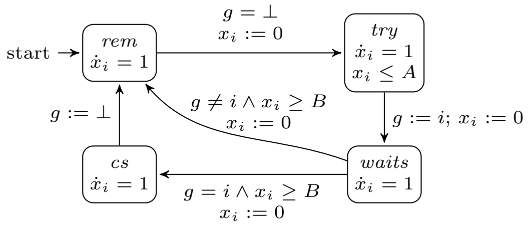


Figure 16: Fischer's mutual exclusion algorithm for a process with identifier $i \in \{1, \ldots, N\}$. Here, g is a global variable of type $\{\bot, 1, \ldots, N\}$, x_i is a local variable of type $\mathbb{R}$, and both A and B are constants of type $\mathbb{R}$.

namely that at most one process in a network of N processes may enter a critical section simultaneously. An automaton for Fischer appears in Fig. 16. Fischer involves two real timing parameters, A and B, and mutual exclusion is ensured iff $A < B$. Let $Loc \triangleq \{rem, try, waits, cs\}$. We translated a network of two automata ($N = 2$) from SpaceEx to SLSF. In one instance, we ensured $A < B$ by picking $A = 5$ and $B = 70$, so mutual exclusion was maintained, which we verified in SpaceEx using the PHAVer scenario. In the other instance, we ensured $A > B$ by picking $A = 75$ and $B = 70$, and mutual exclusion was not maintained. Consequently, we could not verify this instance using SpaceEx's PHAVer scenario since a location $cs \sim cs$ was reachable, corresponding to the case where both processes are in the critical section. We conducted $K = 1000$ simulations with maximum time $T = 1000s$ of the translated SLSF model in each case. In Fig. 17 we show respectively the property satisfaction and violation through the automatic translation from SpaceEx to SLSF by plotting the corresponding locations versus time, where different colors correspond to different simulations. In the safe case ($A < B$), the locations reached via simulations all maintained the mutual exclusion property and were $Loc^2 \setminus \{cs \sim cs, try \sim cs, cs \sim try\}$. In the unsafe case ($A > B$), the locations reached via simulation included every location (e.g., all 16 locations of the permutations of Loc^N for $N = 2$) and violated the mutual exclusion

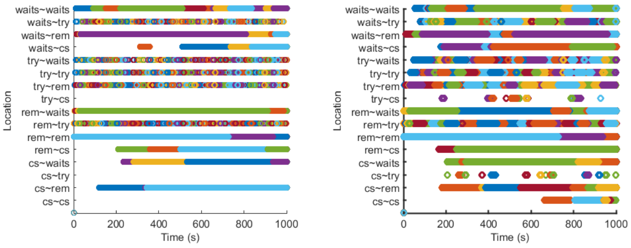


Figure 17: Locations reached for 1000 SLSF simulations of Fischer, where different colors indicate different trajectories. Left: safe case ($A < B$). Right: unsafe case ($A > B$).

property. These results give further empirical evidence for the correctness of the translation procedure.

4.5 Additional Case Studies

Table 1 summarizes the different types of benchmarks that were all successfully translated and checked for trajectory-equivalence in addition to the previously presented case studies. The experiments were performed on an Intel I5 2.4GHz machine with 8GB RAM. All the benchmarks are available in the supplementary material [1].

5 Conclusion

In this paper, we presented a trajectory-equivalent transformation of a hybrid automaton into a continuous-time SLSF diagram, and described its implementation in a prototype software tool. For non-deterministic models, our approach adds auxiliary randomization for various sources of non-determinism to mimic the semantics of hybrid automata. We have empirically validated our approach on a number of challenging benchmarks. To account for zero-crossing issues in the simulation engine, our translation is parametrized by an ε relaxation; for $\varepsilon = 0$ we obtain an under-approximation of the hybrid automaton trajectories (which is precise assuming a perfect simulation engine), while for $\varepsilon > 0$ we obtain an over-approximation.

For the future, it will be interesting to further refine and extend our approach by, e.g., considering the translation of *networks* of hybrid automata—directly without first composing them—into SLSF diagrams and exploring further sources of non-determinism such as non-deterministic flows. Another gainful direction would be to make the distribution over all possible executions uniform. A focus on rare events in the line of [18] could also be considered, and evaluating the SLSF diagrams using tools integrated with SLSF such as S-TaLiRo [5] or Breach [19] would be useful.

Acknowledgment

The material presented in this paper is based upon work supported by the Air Force Office of Scientific Research (AFOSR), in part under contract numbers FA9550-15-1-0258 and W911NF-16-1-0534, by AFRL through contract number FA8750-15-1-0105, and by the National Science Foundation (NSF) under grant numbers CNS 1464311, EPCN 1509804, and CCF 1527398. Furthermore, this research was supported in part by the European Research Council (ERC) under grant 267989 (QUAREM) and by the Austrian Science Fund (FWF) under grant numbers S11402-N23 (RiSE/SHiNE) and Z211-N23 (Wittgenstein Award). Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of AFRL, AFOSR, or NSF.

References

1. Hybrid automata: from verification to implementation – supplementary material, `http://swt.informatik.uni-freiburg.de/tool/spaceex/ha2slsf/ha2slsf`
2. Agrawal, A., Simon, G., Karsai, G.: Semantic translation of Simulink/Stateflow models to hybrid automata using graph transformations. Electronic Notes in Theoretical Computer Science 109, 43–56 (2004)
3. Agut, D.N., van Beek, D., Rooda, J.: Syntax and semantics of the compositional interchange format for hybrid systems. The Journal of Logic and Algebraic Programming 82(1), 1 – 52 (2013)
4. Alur, R., Kanade, A., Ramesh, S., Shashidhar, K.C.: Symbolic analysis for improving simulation coverage of Simulink/Stateflow models. In: Proceedings of the 8th ACM International Conference on Embedded Software. pp. 89–98. EMSOFT '08, ACM, New York, NY, USA (2008)
5. Annpureddy, Y., Liu, C., Fainekos, G., Sankaranarayanan, S.: S-TaLiRo: A tool for temporal logic falsification for hybrid systems. In: TACAS. Springer (2011)
6. Bak, S., Bogomolov, S., Johnson, T.T.: HyST: A source transformation and translation tool for hybrid automaton models. In: HSCC. ACM (2015)
7. Bak, S., Johnson, T.T.: Periodically-scheduled controller analysis using hybrid systems reachability and continuization. In: 36th IEEE Real-Time Systems Symposium (RTSS). IEEE Computer Society, San Antonio, Texas (Dec 2015)
8. Balasubramanian, D., Păsăreanu, C.S., Whalen, M.W., Karsai, G., Lowry, M.: Polyglot: Modeling and analysis for multiple statechart formalisms. In: Proceedings of the 2011 International Symposium on Software Testing and Analysis. pp. 45–55. ISSTA '11, ACM, New York, NY, USA (2011)
9. Bogomolov, S., Donzé, A., Frehse, G., Grosu, R., Johnson, T.T., Ladan, H., Podelski, A., Wehrle, M.: Guided search for hybrid systems based on coarse-grained space abstractions. International Journal on Software Tools for Technology Transfer pp. 1–19 (2015)

No.	Name	Type	$\|Var\|$	$\|Loc\|$	$\|Trans\|$	t_c	t_s
1	biology_1	NLC	7	1	0	8.894	20.912
2	biology_2	NLC	9	1	0	7.892	12.939
3	bouncing_ball	LC	2	1	1	8.149	11.960
4	brusselator	NLC	2	1	0	7.428	10.650
5	buckling_column	NLC	2	1	0	7.738	11.056
6	coupledVanderPol	NLC	4	1	0	8.202	11.746
7	E5	NLC	5	1	0	8.230	36.635
8	fischer_N2_flat_safe	LH	6	16	82	20.158	54.145
9	fischer_N2_flat_unsafe	LH	6	16	82	19.287	59.627
10	glycemic_control_1	NLH	5	3	4	8.319	15.385
11	glycemic_control_2	NLH	5	3	4	8.301	15.567
12	glycemic_control_poly1	NLH	4	9	18	10.528	23.938
13	glycemic_control_poly2	NLH	4	6	10	9.237	19.341
14	helicopter	LC	28	1	0	10.096	14.897
15	Hires	NLC	9	1	0	7.912	9.001
16	jet_engine	NLC	2	1	0	7.667	11.816
17	lac_operon	NLC	2	1	0	7.586	13.257
18	lorentz	NLC	3	1	0	7.739	11.253
19	lotka_volterra	NLC	2	1	0	7.740	11.025
20	circuits_n2	NLH	3	3	2	9.39	13.895
21	circuits_n4	NLH	5	3	2	8.506	14.202
22	circuits_n6	NLH	7	3	2	8.585	15.113
23	circuits_n8	NLH	9	3	2	8.624	15.386
24	circuits_n10	NLH	11	3	2	8.752	15.813
25	circuits_n12	NLH	13	3	2	9.604	19.837
26	OREGO	NLC	4	1	0	9.157	11.111
27	randgen	LH	3	3	6	9.056	15.112
28	Rober	NLC	4	1	0	8.266	16.999
29	roessler	NLC	3	1	0	9.144	12.771
30	small_circuit	NLC	5	1	0	10.265	13.660
31	spiking_neuron	NLH	2	2	2	8.703	13.559
32	spring_pendulum	NC	4	1	0	9.861	6.251
33	vanderpol	NLC	2	1	0	8.119	12.226

Table 1: Overview of the benchmark problems successfully translated to SLSF by using the method in this paper. Column Type presents different classes of dynamics, where LC, NLC, LH, and NLH are abbreviations for linear continuous, nonlinear continuous, linear hybrid, and nonlinear hybrid, respectively. Columns $|Var|$, $|Loc|$, and $|Trans|$ show the number of variables, locations, and transitions, respectively, while t_c and t_s show respectively the time our tool required to translate the model, and the time to simulate the translated SLSF diagram twice.

10. Bogomolov, S., Frehse, G., Greitschus, M., Grosu, R., Pasareanu, C.S., Podelski, A., Strump, T.: Assume-guarantee abstraction refinement meets hybrid systems. In: HVC. pp. 116–131. LNCS, Springer (2014)
11. Bogomolov, S., Frehse, G., Grosu, R., Ladan, H., Podelski, A., Wehrle, M.: A box-based distance between regions for guiding the reachability analysis of SpaceEx. In: CAV. LNCS, vol. 7358, pp. 479–494. Springer (2012)
12. Bogomolov, S., Schilling, C., Bartocci, E., Batt, G., Kong, H., Grosu, R.: Abstraction-based parameter synthesis for multiaffine systems. In: HVC. LNCS, vol. 9434, pp. 19–35. Springer (2015)
13. Bouissou, O., Chapoutot, A.: An operational semantics for Simulink's simulation engine. In: Proceedings of the 13th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, Tools and Theory for Embedded Systems. pp. 129–138. LCTES '12, ACM, New York, NY, USA (2012)
14. Carloni, L., Di Benedetto, M.D., Pinto, A., Sangiovanni-Vincentelli, A.: Modeling techniques, programming languages, design toolsets and interchange formats for hybrid systems. Tech. rep. (2004)
15. Carloni, L.P., Passerone, R., Pinto, A., Sangiovanni-Vincentelli, A.L.: Languages and tools for hybrid systems design. Foundations and Trends in Electronic Design Automation 1 (2006)
16. Chen, M., Ravn, A.P., Wang, S., Yang, M., Zhan, N.: A two-way path between formal and informal design of embedded systems. In: UTP. Lecture Notes in Computer Science, vol. 10134, pp. 65–92. Springer (2016)
17. Chen, X., Abraham, E., Sankaranarayanan, S.: Flow*: An analyzer for non-linear hybrid systems. In: Sharygina, N., Veith, H. (eds.) CAV, LNCS, vol. 8044, pp. 258–263. Springer Berlin Heidelberg (2013)
18. Clarke, E.M., Zuliani, P.: Statistical model checking for cyber-physical systems. In: Bultan, T., Hsiung, P.A. (eds.) Automated Technology for Verification and Analysis, LNCS, vol. 6996, pp. 1–12. Springer (2011)
19. Donzé, A.: Breach, a toolbox for verification and parameter synthesis of hybrid systems. In: Touili, T., Cook, B., Jackson, P. (eds.) Computer Aided Verification, LNCS, vol. 6174, pp. 167–170. Springer Berlin / Heidelberg (2010)
20. Duggirala, P.S., Mitra, S., Viswanathan, M.: Verification of annotated models from executions. In: Proceedings of the Eleventh ACM International Conference on Embedded Software (EMSOFT '13). IEEE Press, Piscataway, NJ, USA (2013)
21. Fisher, M.E.: A semiclosed-loop algorithm for the control of blood glucose levels in diabetics. Biomedical Engineering, IEEE Transactions on 38(1), 57–61 (1991)
22. Frehse, G., Le Guernic, C., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., Maler, O.: SpaceEx: Scalable verification of hybrid systems. In: Computer Aided Verification. pp. 379–395 (2011)
23. Hamon, G.: A denotational semantics for Stateflow. In: Proceedings of the 5th ACM International Conference on Embedded Software. pp. 164–172. EMSOFT '05, ACM, New York, NY, USA (2005)
24. Hamon, G., Rushby, J.: An operational semantics for Stateflow. International Journal on Software Tools for Technology Transfer 9(5-6), 447–456 (2007)
25. Jiang, Z., Pajic, M., Alur, R., Mangharam, R.: Closed-loop verification of medical devices with model abstraction and refinement. International Journal on Software Tools for Technology Transfer 16(2), 191–213 (2014)
26. Johansson, K.H., Egerstedt, M., Lygeros, J., Sastry, S.: On the regularization of zeno hybrid automata. Systems & Control Letters 38(3), 141–150 (1999)
27. Larsen, K.G., Pettersson, P., Yi, W.: UPPAAL in a nutshell. International Journal on Software Tools for Technology Transfer (STTT) 1(1), 134–152 (1997)
28. Lavalle, S.M., Kuffner, J.J., Jr.: Rapidly-exploring random trees: Progress and prospects. In: Algorithmic and Computational Robotics: New Directions. pp. 293–308 (2000)
29. Manamcheri, K., Mitra, S., Bak, S., Caccamo, M.: A step towards verification and synthesis from Simulink/Stateflow models. In: Proc. of the 14th Intl. Conf. on Hybrid Systems: Computation and Control (HSCC). pp. 317–318. ACM (2011)
30. Minopoli, S., Frehse, G.: From simulation models to hybrid automata using urgency and relaxation. In: Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control. pp. 287–296. HSCC '16, ACM, New York, NY, USA (2016)
31. Minopoli, S., Frehse, G.: SL2SX translator: From Simulink to SpaceEx models. In: Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control. pp. 93–98. HSCC '16, ACM, New York, NY, USA (2016)
32. Nguyen, L.V., Johnson, T.T.: Benchmark: DC-to-DC switched-mode power converters (buck converters, boost converters, and buck-boost converters). In: Applied Verification for Continuous and Hybrid Systems Workshop (ARCH 2014). Berlin, Germany (Apr 2014)
33. Pajic, M., Mangharam, R., Sokolsky, O., Arney, D., Goldman, J., Lee, I.: Model-driven safety analysis of closed-loop medical systems. Industrial Informatics, IEEE Transactions on 10(1), 3–16 (2014)
34. Pajic, M., Jiang, Z., Lee, I., Sokolsky, O., Mangharam, R.: From verification to implementation: A model translation tool and a pacemaker case study. In: Real-Time and Embedded Technology and Applications Symposium (RTAS), 2012 IEEE 18th. pp. 173–184. IEEE (2012)
35. Pajic, M., Jiang, Z., Lee, I., Sokolsky, O., Mangharam, R.: Safety-critical medical device development using the UPP2SF model translation tool. ACM Trans. Embed. Comput. Syst. 13(4s), 127:1–127:26 (Apr 2014)
36. Pinto, A., Carloni, L., Passerone, R., Sangiovanni-Vincentelli, A.: Interchange format for hybrid systems: Abstract semantics. In: Hespanha, J.P., Tiwari, A. (eds.) Hybrid Systems: Computation and Control, LNCS, vol. 3927, pp. 491–506. Springer Berlin Heidelberg (2006)
37. Pinto, A., Sangiovanni-Vincentelli, A.L., Carloni, L.P., Passerone, R.: Interchange formats for hybrid systems: Review and proposal. In: Morari, M., Thiele, L. (eds.) Hybrid Systems: Computation and Control, LNCS, vol. 3414, pp. 526–541. Springer Berlin Heidelberg (2005)
38. Sampath, P., Rajeev, A.C., Ramesh, S.: Translation validation for Stateflow to C. In: Proceedings of the The 51st Annual Design Automation Conference on Design Automation Conference. pp. 23:1–23:6. DAC '14, ACM, New York, NY, USA (2014)

39. Sanfelice, R., Copp, D., Nanez, P.: A toolbox for simulation of hybrid systems in Matlab/Simulink: Hybrid equations (HyEQ) toolbox. In: Proceedings of the 16th international conference on Hybrid systems: computation and control. pp. 101–106. ACM (2013)
40. Schrammel, P., Jeannet, B.: From hybrid data-flow languages to hybrid automata: A complete translation. In: Proceedings of the 15th ACM International Conference on Hybrid Systems: Computation and Control. pp. 167–176. HSCC '12, ACM, New York, NY, USA (2012)
41. Severns, R.P., Bloom, G.: Modern DC-to-DC Switchmode Power Converter Circuits. Van Nostrand Reinhold Company, New York, New York (1985)
42. Tiwari, A., Shankar, N., Rushby, J.: Invisible formal methods for embedded control systems. Proceedings of the IEEE 91(1), 29–39 (Jan 2003)
43. Yan, G., Jiao, L., Li, Y., Wang, S., Zhan, N.: Approximate bisimulation and discretization of hybrid CSP. In: FM. Lecture Notes in Computer Science, vol. 9995, pp. 702–720 (2016)
44. Zou, L., Zhan, N., Wang, S., Fränzle, M.: Formal verification of simulink/stateflow diagrams. In: ATVA. Lecture Notes in Computer Science, vol. 9364, pp. 464–481.
Springer (2015)

Charge Pump Phase-Locked Loops and Full Wave Rectifiers for Reachability Analysis (Benchmark Proposal)

Omar Ali Beg, Ali Davoudi, and Taylor T. Johnson

University of Texas at Arlington, USA

Abstract

Analog-mixed signal (AMS) circuits are widely used in various mission-critical applications necessitating their formal verification prior to implementation. We consider modeling two AMS circuits as hybrid automata, particularly a charge pump phase-locked loop (CP-PLL) and a full-wave rectifier (FWR). We present executable models for the benchmarks in SpaceEx format, perform reachability analysis, and demonstrate their automatic conversion to MathWorks Simulink/Stateflow (SLSF) format using the HyST tool. Moreover, as a next step towards implementation, we present the VHDL-AMS description of a circuit based on the verified model.

Category: academic **Difficulty:** medium

1 Context and Origins

Many analog-mixed signal (AMS) circuits are widely used in various mission cicritical applications and require formal verification prior implementation. Formal verification methods construct a mathematical model $\mathcal{M}$ with precise semantics, provide extensive analysis with respect to some correctness requirement $\mathcal{P}$, and verify that $\mathcal{M} \models \mathcal{P}$ [2]. This can be ascertained through reachability analysis [1]. As an example of circuitry that can benefit from formal verification prior to field implementation and deployment, we provide two potential benchmarks for hybrid verification research community, i.e., charge pump phase-locked loop (CP-PLL), and full-wave rectifier (FWR).

CP-PLL integrated circuits are widely used in modern mobile, radio, and wireless communication applications to synchronize a high-frequency signal with a low-frequency reference signal. In [8], the auhtors use SpaceEx model checking tool [6] to verify the global convergence with respect to phase and frequency lock for a digital PLL. An FWR converts an AC electric input signal to a DC output signal, and formal verification through reachability analysis has been reported using different model checking tools in [5], except SpaceEx. We develop hybrid automaton models of CP-PLL and FWR, and used SpaceEx [6], a reachability analysis tool, to compute the over-approximated sets of reachable states [1]. This a classical fixed point computation tool that operates on symbolic states.

We also use HyST (Hybrid Source Transformer) [3] to automatically convert the hybrid automaton models developed in SpaceEx to MathWorks Simulink/Stateflow (SLSF) models [2]. It is a source-to-source translation tool that takes input in the SpaceEx model format, and translates it to the formats of HyCreate,Flow*, dReach, C2E2, Passel 2.0, and HyComp. Additional tool support is being added from time to time. Verification and validation research community may use HyST to automatically transform the hybrid automaton models in SpaceEx format to

[1] The tool is available online from the SpaceEx website at: http://spaceex.imag.fr/.

[2] The executable models are included on the ARCH website and are also available online from the HyST website at: http://verivital.com/hyst/.

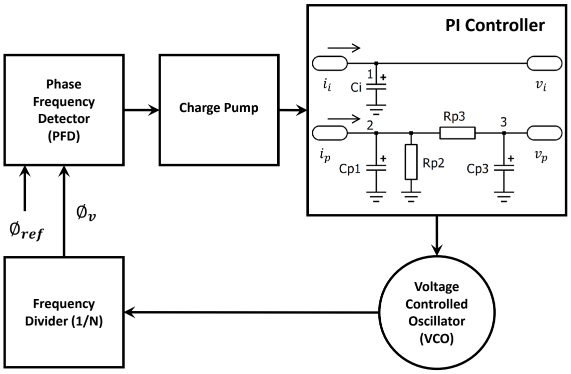


Figure 2.1: Block diagram of the PLL circuit with a PI controller.

other formats and perform reachability analysis using aforesaid model checking tools. Finally, we present VHDL-AMS description of an FWR.

2 Hybrid Automaton Modeling of CP-PLL and FWR

In this section, we present the hybrid automaton modeling of CP-PLL and FWR.

2.1 CP-PLL Modeling

We consider a third-order CP-PLL as described in [1]. It consists of a reference frequency signal generator, a phase frequency detector (PFD), a charge pump, a proportional-integral (PI) controller, a voltage- controlled oscillator (VCO) and a frequency divider as shown in Figure 2.1. The state variables are defined by the voltages across the capacitors C_i, C_{p1}, and C_{p3}, i.e., v_i, v_{p1}, and v_p respectively. Two more state variables are defined by the dynamics of VCO and reference frequencies, i.e., ϕ_v and ϕ_{ref}, respectively. CP-PLL is designed such that ϕ_v locks on to ϕ_{ref}, that may constitute the property of CP-PLL to be verified. This locking is ensured by PFD using the phase difference of ϕ_{ref} and ϕ_v to generate 'UP' or 'DN' signal for the charge pump.

The ODEs from the CP-PLL circuit diagram can be readily formed using the traditional circuit analysis techniques, i.e., Kirchoff's voltage law (KVL) and Kirchoff's current law (KCL). We apply KCL at node 1 of the circuit used to implemented the analog PI controller shown in Figure 2.1

$$i_i = i_{C_i} \tag{2.1}$$

We can write the above equation in terms of voltage across capacitor C_i as

$$C_i.\dot{v}_i = i_i. \tag{2.2}$$

Rearranging the above equation, we obtain

$$\dot{v}_i = \frac{i_i}{C_i} \tag{2.3}$$

We apply KCL at node 2 of the the circuit used to implemented the analog PI controller in Figure 2.1 to get

$$i_p = i_{C_{p1}} + i_{R_{p2}} + i_{R_{p3}} \tag{2.4}$$

Replacing the current terms with voltage terms in right hand side of above equation, we get

$$i_p = C_{p1}\dot{v}_{p1} + \frac{v_{p1}}{R_{p2}} + \frac{(v_{p1} - v_p)}{R_{p3}}. \tag{2.5}$$

Rearranging the above equation for $\dot{v}_{p1}$, we get

$$\dot{v}_{p1} = -\frac{v_{p1}}{C_{p1}}\left(\frac{1}{R_{p2}} + \frac{1}{R_{p3}}\right) + \frac{v_p}{C_{p1}R_{p3}} + \frac{i_p}{C_{p1}}. \tag{2.6}$$

Next, we may apply KCL at node 3 to get

$$i_C p3 = i_{R_p 3} \tag{2.7}$$

Re-writing the above equation in terms of voltages, we get

$$C_{p3}\dot{v}_p = \frac{v_{p1} - v_p}{R_{p3}} \tag{2.8}$$

Rearranging the above equation leads to

$$\dot{v}_p = \frac{v_{p1}}{C_{p3}R_{p3}} - \frac{v_p}{C_{p3}R_{p3}}. \tag{2.9}$$

For the VCO, the output phase ϕ_v is the integral of the frequency and the input voltages, i.e., v_i, and v_p [7]. We also include the frequency division factor N to obtain the ODE as

$$\dot{\phi}_v = \frac{K_i}{N}v_i + \frac{K_p}{N}v_p + \frac{2\pi}{N}f_0 \tag{2.10}$$

and

$$\dot{\phi}_{ref} = 2\pi f_{ref}. \tag{2.11}$$

Here, K_i and K_p are the voltage-to-frequency gains for v_i and v_p respectively, and f_0 is the frequency of VCO. These ODEs depict the continuous dynamics within each discrete location. The input to the PI controller, i.e., $[i_i, i_p]^T$, is generated by the charge pump depending upon the relative phase of ϕ_v and ϕ_{ref}. This phase difference is measured by PFD, which generates an 'UP' signal if ϕ_{ref} leads ϕ_v, and 'DN' signal if ϕ_v leads ϕ_{ref}. An 'UP' signal will charge the capacitors, hence increasing the voltages across the capacitors of the proportional and integrator channel, i.e., v_p and v_i, respectively, leading to an increased VCO frequency. On the other hand, a 'DN' signal from PFD will tend the charge pump to produce current in reverse direction to discharge the capacitors, hence reducing the voltages in the PI channel. The reduced v_p and v_i voltages will result in a reduced ϕ_v to make it track ϕ_{ref}. Depending upon the status of Up/Down signals, there may be four discrete locations (i.e., the input varies for each discrete location) as follows:

1 $Both_0$ (i.e. Both OFF): The input vector is given by $[i_i, i_p]^T = [0, 0]^T$

2 Up_1 (i.e. UP ON): The input vector is given by $[i_i, i_p]^T = \left[I_i^{up}, I_p^{up}\right]^T$

3 $Both_1$ (i.e. Both ON): The input vector is given by $[i_i, i_p]^T = \left[I_i^{up} + I_i^{dn}, I_p^{up} + I_p^{dn}\right]^T$

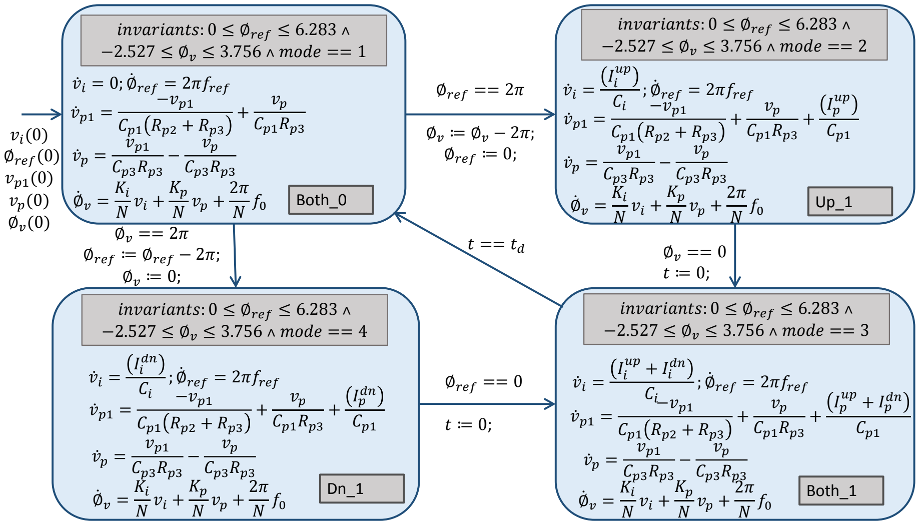

invariants: $0 \le \emptyset_{ref} \le 6.283 \wedge -2.527 \le \emptyset_v \le 3.756 \wedge mode == 1$
$\dot{v}_i = 0; \dot{\emptyset}_{ref} = 2\pi f_{ref}$
$\dot{v}_{p1} = \frac{-v_{p1}}{C_{p1}(R_{p2}+R_{p3})} + \frac{v_p}{C_{p1}R_{p3}}$
$\dot{v}_p = \frac{v_{p1}}{C_{p3}R_{p3}} - \frac{v_p}{C_{p3}R_{p3}}$
$\dot{\emptyset}_v = \frac{K_i}{N}v_i + \frac{K_p}{N}v_p + \frac{2\pi}{N}f_0$
Both_0
$v_i(0)$
$\emptyset_{ref}(0)$
$v_{p1}(0)$
$v_p(0)$
$\emptyset_v(0)$
$\emptyset_{ref} == 2\pi$
$\emptyset_v := \emptyset_v - 2\pi;$
$\emptyset_{ref} := 0;$
invariants: $0 \le \emptyset_{ref} \le 6.283 \wedge -2.527 \le \emptyset_v \le 3.756 \wedge mode == 2$
$\dot{v}_i = \frac{(I_i^{up})}{C_i}; \dot{\emptyset}_{ref} = 2\pi f_{ref}$
$\dot{v}_{p1} = \frac{-v_{p1}}{C_{p1}(R_{p2}+R_{p3})} + \frac{v_p}{C_{p1}R_{p3}} + \frac{(I_p^{up})}{C_{p1}}$
$\dot{v}_p = \frac{v_{p1}}{C_{p3}R_{p3}} - \frac{v_p}{C_{p3}R_{p3}}$
$\dot{\emptyset}_v = \frac{K_i}{N}v_i + \frac{K_p}{N}v_p + \frac{2\pi}{N}f_0$
Up_1
$\emptyset_v == 2\pi$
$\emptyset_{ref} := \emptyset_{ref} - 2\pi;$
$\emptyset_v := 0;$
$t == t_d$
$\emptyset_v == 0$
$t := 0;$
invariants: $0 \le \emptyset_{ref} \le 6.283 \wedge -2.527 \le \emptyset_v \le 3.756 \wedge mode == 4$
$\dot{v}_i = \frac{(I_i^{dn})}{C_i}; \dot{\emptyset}_{ref} = 2\pi f_{ref}$
$\dot{v}_{p1} = \frac{-v_{p1}}{C_{p1}(R_{p2}+R_{p3})} + \frac{v_p}{C_{p1}R_{p3}} + \frac{(I_p^{dn})}{C_{p1}}$
$\dot{v}_p = \frac{v_{p1}}{C_{p3}R_{p3}} - \frac{v_p}{C_{p3}R_{p3}}$
$\dot{\emptyset}_v = \frac{K_i}{N}v_i + \frac{K_p}{N}v_p + \frac{2\pi}{N}f_0$
Dn_1
$\emptyset_{ref} == 0$
$t := 0;$
invariants: $0 \le \emptyset_{ref} \le 6.283 \wedge -2.527 \le \emptyset_v \le 3.756 \wedge mode == 3$
$\dot{v}_i = \frac{(I_i^{up}+I_i^{dn})}{C_i}; \dot{\emptyset}_{ref} = 2\pi f_{ref}$
$\dot{v}_{p1} = \frac{-v_{p1}}{C_{p1}(R_{p2}+R_{p3})} + \frac{v_p}{C_{p1}R_{p3}} + \frac{(I_p^{up}+I_p^{dn})}{C_{p1}}$
$\dot{v}_p = \frac{v_{p1}}{C_{p3}R_{p3}} - \frac{v_p}{C_{p3}R_{p3}}$
$\dot{\emptyset}_v = \frac{K_i}{N}v_i + \frac{K_p}{N}v_p + \frac{2\pi}{N}f_0$
Both_1

Figure 2.2: Hybrid automaton model for CP-PLL system.

4 Dn_1 (i.e. DN ON): The input vector is given by $[i_i, i_p]^T = [I_i^{dn}, I_p^{dn}]^T$

Accordingly, using the above ODEs and the inputs defined, a hybrid automaton is shown in Figure 2.2. The component values used in the model are as per Table 1 of [1]. Moreover, the input values are: $I_i^{up} = 10.1\mu A, I_i^{dn} = -10.1\mu A$, and $I_p^{up} = 505\mu A, I_p^{dn} = -505\mu A$. The guard conditions for discrete transitions are formed depending upon ϕ_{ref} and ϕ_v. As discussed earlier, the PFD output depends on whether ϕ_{ref} leads or lags with respect to ϕ_v. If the initial discrete location is $Both_0$, the automaton jumps to Up_1 if ϕ_{ref} leads as $\phi_{ref} = 2\pi$, otherwise it jumps to Dn_1 if ϕ_v leads as $\phi_v = 2\pi$. There is a design requirement to introduce a time delay, t_d, required to switch off both the charge pumps. This is represented by the location $Both_1$. Once the lagging signal reaches zero, the automaton jumps to this location and, once $t = t_d$, the automaton transitions back to $Both_0$.

2.2 FWR Modeling

We consider an FWR as described in [5]. It is basically a full-wave diode bridge, that consists of two diodes D_1 and D_2, a capacitor C and the load resistor R as shown in Figure 2.3. An AC input signal is supplied to the circuit through a center-tapped transformer. For the modeling purpose, and without the lack of generality, we use two AC sources as shown in Figure 2.3. This circuit converts the input AC voltage V_{in} to a DC voltage V_o, at its output measured across R. We may need to verify that V_o is stable within $\pm 1\% V_{max}$ for the steady-state operation, where V_{max} is the maximum value of the input AC signal.

For modeling purposes, we consider R_d as the forward resistance of each diode. Let the current through R_d, C, and R be i_{Rd}, i_C, and i_R, respectively. The input sinusoidal voltage be $V_{in} = V_{max} sin(2\pi ft)$, and the output voltage across the load resistor R be V_o, where, V_{max} is the maximum amplitude of the sinusoidal signal and f is its frequency. For model checking purposes, we use SpaceEx that requires hybrid automaton model with linear dynamics, so we model the input AC signal using a second-order differential equation [5]. We define another

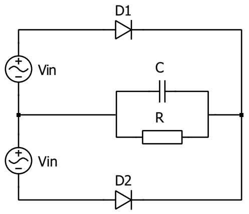


Figure 2.3: Schematic diagram of FWR.

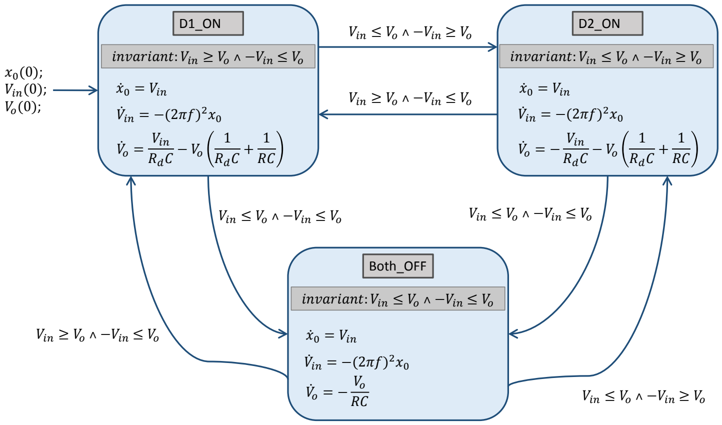

D1_ON
invariant: $V_{in} \geq V_o \wedge -V_{in} \leq V_o$
$\dot{x}_0 = V_{in}$
$\dot{V}_{in} = -(2\pi f)^2 x_0$
$\dot{V}_o = \frac{V_{in}}{R_d C} - V_o\left(\frac{1}{R_d C} + \frac{1}{RC}\right)$
$x_0(0)$;
$V_{in}(0)$;
$V_o(0)$;
$V_{in} \leq V_o \wedge -V_{in} \geq V_o$
$V_{in} \geq V_o \wedge -V_{in} \leq V_o$
D2_ON
invariant: $V_{in} \leq V_o \wedge -V_{in} \geq V_o$
$\dot{x}_0 = V_{in}$
$\dot{V}_{in} = -(2\pi f)^2 x_0$
$\dot{V}_o = -\frac{V_{in}}{R_d C} - V_o\left(\frac{1}{R_d C} + \frac{1}{RC}\right)$
$V_{in} \leq V_o \wedge -V_{in} \leq V_o$
$V_{in} \leq V_o \wedge -V_{in} \leq V_o$
Both_OFF
invariant: $V_{in} \leq V_o \wedge -V_{in} \leq V_o$
$\dot{x}_0 = V_{in}$
$\dot{V}_{in} = -(2\pi f)^2 x_0$
$\dot{V}_o = -\frac{V_o}{RC}$
$V_{in} \geq V_o \wedge -V_{in} \leq V_o$
$V_{in} \leq V_o \wedge -V_{in} \geq V_o$

Figure 2.4: Hybrid automaton model for FWR system.

state variable x_0 and model the AC input by ODEs defined as

$$\dot{x}_0 = V_{in} \tag{2.12}$$

and

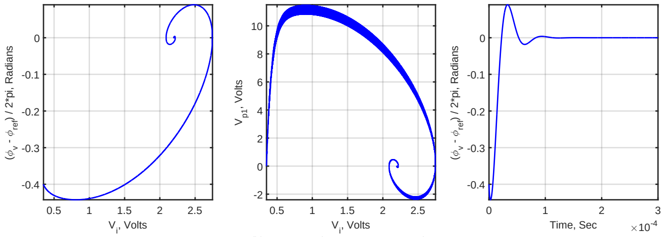
$$\dot{V}_{in} = -\left(2\pi f\right)^2 x_0 \tag{2.13}$$

The solution of above system is $V_{in} = V_{max} sin(2\pi f t)$ such that the initial conditions are $x_0 = \dfrac{-V_{max}}{2\pi f}$ and $V_{in} = 0$. Next, we consider the FWR circuit dynamics to form ODE for V_o. The circuit dynamics depend upon the operation of diodes D_1 and D_2. Accordingly, we may form three different topological instances, i.e., D_1 ON and D_2 OFF, D_1 OFF and D_2 ON, and both the diodes OFF when $V_{in} \leq V_o$. There could be a fourth topological instance, i.e., both the diodes ON at the same time, but this is not practical due to the nature of the sinusoidal input. Therefore, we may consider three topologies one by one to form the ODEs and start with the topology with D_1 ON and D_2 OFF. The invariants for this topological instance are $V_{in} \geq V_o \wedge -V_{in} \leq V_o$. Applying KCL at the node joining C and R in Figure 2.3, we get

$$i_{Rd} = i_C + i_R \tag{2.14}$$

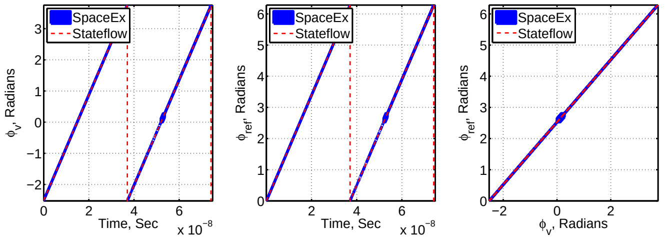
and we can express the above equation in terms of voltages as

$$\frac{V_{in} - V_o}{R_d} = C\dot{V}_o + \frac{V_o}{R}. \tag{2.15}$$



(a) Phase difference vs. Integrator voltage (b) Cap. p_1 Voltage vs. Integrator voltage (c) Phase difference vs. time

Figure 3.1: SLSF plots for PLL showing stable limit cycles and ϕ_v locking onto ϕ_{ref} within 0.2 mSec.



(a) VCO frequency vs. time (b) Ref. frequency vs. time (c) Ref. frequency vs. VCO frequency

Figure 3.2: Comparison of SpaceEx reach sets and SLSF trajectories for PLL.

Rearranging the above equation provides

$$\dot{V_o} = \frac{V_{in}}{R_d C} - V_o \left(\frac{1}{R_d C} + \frac{1}{RC} \right). \tag{2.16}$$

By the same token, for D_1 OFF and D_2 ON with invariants $V_{in} \le V_o \wedge -V_{in} \ge V_o$, we use KCL at the same node in Figure 2.3 to get

$$\dot{V_o} = -\frac{V_{in}}{R_d C} - V_o \left(\frac{1}{R_d C} + \frac{1}{RC} \right). \tag{2.17}$$

For the topology when both D_1 and D_2 are OFF, the sinusoidal input signal is cut off from the entire circuit and the load voltage is only provided by the capacitor. The invariants for this topological instance are $V_{in} \le V_o \wedge -V_{in} \le V_o$. Therefore, we get

$$\dot{V_o} = -\frac{V_o}{RC}. \tag{2.18}$$

Accordingly, the hybrid automaton model of FWR is shown in Figure 2.4. In addition, we consider the VHDL-AMS description of FWR in Section A, where the circuit is externally supplied by V_{in}.

3 SLSF Simulations and Reachability Analysis

Formal verification of CP-PLL constitutes verifying its frequency-locking property, i.e., whether ϕ_v locks onto ϕ_{ref}. For this purpose, we need to compute the phase difference between ϕ_v and

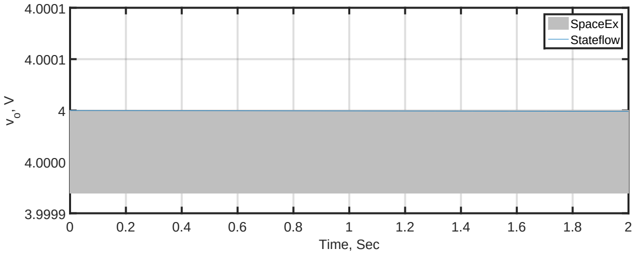


Figure 3.3: Comparison of SpaceEx and SLSF for the output voltage of FWR in the steady state, showing the simulation trace containment within overapproximated sets of reachable states.

ϕ_{ref}. The SLSF plots for the phase difference vs. integrator voltage, voltage of capacitor p_1 vs. integrator voltage v_i, and the phase difference versus time are shown in Figure 3.1. The first two plots depict a stable limit cycle highlighting stability properties of CP-PLL. In the third plot, we show that the phase difference between ϕ_{ref} and ϕ_v reaches zero within 0.2 mSec., signifying that ϕ_v locks onto ϕ_{ref} within such time intervals.

We also analyze the hybrid automaton using SpaceEx, and a comparison of the first few iterations for SpaceEx and SLSF is shown in Figure 3.2. We show that SLSF simulation traces, and the over-approximated sets of reachable states computed using SpaceEx, match for the first five iterations. CP-PLL requires thousands of cycles to lock, hence there will be thousands of discrete transitions for the switching logic resulting inaccuracy due to SpaceEx overapproximations [1]. It is evident from comparing the first five iterations in Figure 3.2 that SLSF simulation traces are contained within the over-approximated sets of reachable states. We also conclude that the SLSF traces exhibit stable limit cycles, and that frequency locking is achieved within 0.2 mSec.

As evident from this benchmark, the performance of reachability analysis tools is not satisfactory due to the high number of discrete transitions (practically being in order of thousands). It is pertinent to highlight that in [4], the authors have used a variant of continuization [1] to address this problem for the design of a yaw damper system for a 747 jet aircraft. Continuization is a process whereby the abstraction of a hybrid system having large number of discrete transitions is obtained by a continuous system with an extra non-deterministic input. The authors use HyST to automatically transform the model and perform reachability analysis using Flow* and SpaceEx to display satisfactory results in [4]. A similar approach can be used for this benchmark so as to perform reachability analysis using SpaceEx and Flow*.

We perform the reachability analysis using SpaceEx under the steady-state conditions for FWR, i.e., $V_{max} = 4V$, $V_o(0) = 4V$, and $f = 50Hz$, as shown in Figure 3.3. The steady-state SLSF time traces for the output voltage are contained within the over-approximated sets of reachable states computed using SpaceEx.

During conversion from SpaceEx to SLSF using HyST, the conversion time noted for CP-PLL is 1.633077 seconds and that for FWR is 1.936676 seconds. We used MATLAB Release 2015a on a Windows 7, 64 bit operating system with Intel Core i7-2600 CPU at 3.40 GHz and 16 GB RAM.

4 Key Observations

Hybrid automaton modeling and reachability analysis of CP-PLL using traditional model checking tools, such as SpaceEx, is an extensive challenge. This is due to the reason that CP-PLL requires thousand of cycles to lock, resulting in thousand of discrete transitions in the switching logic. Therefore, the SpaceEx analysis did not produce accurate reachability results if the analysis is run for an extended duration of time. This requires some advanced techniques, such as continuization [1] that is demostarted in [4] using HyST, SpaceEx, and Flow*. For FWR, SpaceEx produced a run-time error due to non-affine dynamics as the model had pure sinusoidal time-dependent signal as an input. Therefore, we have modeled the sinusoidal input signal using the second-order ODEs to successfully compute the reachability analysis results.

5 Benchmark Outlook

Overall, these verification benchmarks have medium difficulty level, and can serve as a first step towards a benchmark library to evaluate reachability and verification methods for AMS circuits. These benchmarks are open to the continuous and hybrid systems verification community to evaluate their methods and tools.

Acknowledgments The material presented in this paper is based upon work supported by the National Science Foundation (NSF) under grant numbers CNS 1464311 and CCF 1527398, the Air Force Research Laboratory (AFRL) through contract number FA8750-15-1-0105, and the Air Force Office of Scientific Research (AFOSR) under contract number FA9550-15-1-0258. The U.S. government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of AFRL, AFOSR, or NSF.

References

[1] Matthias Althoff, Akshay Rajhans, Bruce H. Krogh, Soner Yaldiz, Xin Li, and Larry Pileggi. Formal verification of phase-locked loops using reachability analysis and continuization. *Commun. ACM*, 56(10):97–104, October 2013.

[2] Rajeev Alur. Formal verification of hybrid systems. In *Embedded Software (EMSOFT), 2011 Proceedings of the International Conference on*, pages 273–278. IEEE, 2011.

[3] Stanley Bak, Sergiy Bogomolov, and Taylor T Johnson. Hyst: a source transformation and translation tool for hybrid automaton models. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, pages 128–133. ACM, 2015.

[4] Stanley Bak and Taylor T. Johnson. Periodically-scheduled controller analysis using hybrid systems reachability and continuization. In *36th IEEE Real-Time Systems Symposium (http://2015.rtss.org/RTSS 2015/)*, San Antonio, Texas, December 2015. IEEE Computer Society.

[5] Luca P Carloni, Roberto Passerone, Alessandro Pinto, and Alberto Sangiovanni-Vincentelli. *Languages and tools for hybrid systems design.* now Publishers Inc, 2006.

[6] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. Spaceex: Scalable verification of hybrid systems. In Shaz Qadeer Ganesh Gopalakrishnan, editor, *Proc. 23rd International Conference on Computer Aided Verification (CAV)*, LNCS. Springer, 2011.

[7] Jaewook Kim and Seonghwan Cho. A time-based analog-to-digital converter using a multi-phase voltage controlled oscillator. In *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, pages 3934–3937, May 2006.

[8] Jijie Wei, Yan Peng, Ge Yu, and M. Greenstreet. Verifying global convergence for a digital phase-locked loop. In *Formal Methods in Computer-Aided Design (FMCAD), 2013*, pages 113–120, Oct 2013.

A Appendix: VHDL-AMS Description of FWR

As discussed in Section 2, the FWR circuit behavior depends upon the state of the diodes being ON or OFF due to the input sinusoidal signal. We assume that this signal is supplied externally, and form the description as per Equation 2.16, Equation 2.17, and Equation 2.18. It should be mentioned that, in VHDL-AMS, we must minimize the use of the division operation. VHDL-AMS models are typically comprised of two sections, i.e., an entity and an architecture. Entity describes the model interface to the outside world, whereas, architecture describes the function or behavior of the model. A VHDL-AMS description is given below:

```
library ieee;
use ieee.electrical_systems.all;
use ieee.math_real.all;
entity fwr is
   port ( terminal input: electrical;
          terminal output: electrical );
end entity fwr;
----------------------------------------------------------------
architecture dot of fwr is
   quantity vin across input to electrical_ref;
   quantity vout across output to electrical_ref;
   constant r : real := 1000; -- load resistance
   constant rd : real := 0.1; -- diode forward resistance
   constant cap : real := 0.001; -- capacitance
begin
   if vin >= vout and -vin <= vout use
      vin == vout’dot * r * rd + vout + vout * rd / r; -- diode D1 ON
   elseif vin <= vout and -vin >= vout use
      - vin == vout’dot * r * rd + vout + vout * rd / r; -- diode D2 ON
   elseif vin <= vout and - vin <= vout use
      vout == - vout’dot * r * cap; -- Both OFF
   end if;
end architecture dot;
```

Reachability Analysis of Transformer-Isolated DC-DC Converters (Benchmark Proposal)

Omar A. Beg[1], Ali Davoudi[1], and Taylor T. Johnson[2]

[1] University of Texas at Arlington, Arlington, Texas, USA
omar.beg@mavs.uta.edu, davoudi@uta.edu
[2] Vanderbilt University, Nashville, Tennessee, USA
taylor.johnson@vanderbilt.edu

Abstract

Various mission-critical applications necessarily require a transformer in switching converters to obtain DC isolation between the converters' input and output. Since DC-DC converters are the switching devices, these are modeled as hybrid automata. We present hybrid automaton modeling of two main types of transformer isolated DC-DC converters, namely, flyback and forward converters. We have also catered the non-determinism for both. We use HyST (Hybrid Source Transformation) tool to automatically generate the models in SpaceEx format, perform reachability analysis, and then automatically convert the models into Mathworks Simulink Stateflow (SLSF) using HyST. Thus we demonstrate effectiveness of HyST tool in the model-based design process. The HyST user needs not to manually construct or modify the models thus saving significant amount of time and efforts.

Category: academic **Difficulty:** medium

1 Context and Origins

DC-DC converters are the power electronics devices that are extensively used in automotives, industrial, and defense related applications and their mission-critical nature necessitates formal verification prior implementation. Over the period, there has been a drastic rise in power electronics-related safety recalls in the automotive industry. For example, the main cause for recall of around 700,000 Toyota Prius cars in 2014 was attributed to an error in the interaction between a boost converter and its software controller [11]. Likewise, more than 100,000 Toyota Prius cars were recalled due to an inverter failure [12]. Therefore, this mission-critical domain would require significant confidence in the modeling accuracy. This can be ensured through reachability analysis [1, 6, 7]. We present two potential benchmarks related to transformer-isolated DC-DC converters for hybrid verification research community. Transformer isolation is implemented by introducing a transformer at the converter input. In addition to the electrical isolation between the input and the output, transformer-isolated DC-DC converters have some other advantages compared to their non-isolated counterparts such as high efficiency and low manufacturing cost [4]. Due to their advantages, these are preferred for the DC-DC applications in industrial and defense-related control/communication systems and distributed power networks. This work is based on hybrid automaton modeling of two main types of transformer-isolated DC-DC converters, i.e., flyback converter and forward converter. This is a series of benchmarks [6–8] that are being developed to benefit from formal verification prior to field implementation and deployment.

Flyback converter may be regarded as a transformer-isolated buck-boost converter, whereas, forward converter acts as a transformer-isolated buck converter. We develop hybrid automaton models of flyback and forward converters, and use SpaceEx [5], a reachability analysis tool, to compute the over-approximated sets of reachable states [1]. This is a classical fixed point computation tool that operates on symbolic states.

We also use HyST (Hybrid Source Transformation) tool [2] to automatically convert the hybrid automaton models developed in SpaceEx to MathWorks Simulink/Stateflow (SLSF) models [2]. It is a source-to-source translation tool that takes input in the SpaceEx model format, and translates it to the formats of HyCreate, Flow*, dReach, C2E2, Passel 2.0, and HyComp. In addition, it is also used to automatically generate the hybrid automaton models in SpaceEx format as per user-defined parameters and settings. Additional tool support is being added from time to time. Verification and validation research community may use HyST to automatically transform the hybrid automaton models in SpaceEx format to other formats and perform reachability analysis using aforesaid model checking tools.

2 Hybrid Automaton Modeling of Transformer-Isolated DC-DC Converters

We present the hybrid automaton modeling of flyback and forward converters in this section. We assume that transformer losses are negligible with perfect coupling among the windings. The transformer is modeled using a parallel magnetizing inductance L_m at the input side, called the primary side. The winding towards the output is called the secondary winding. Let n be the turns ratio of primary to secondary windings. Let v_1 and v_2 be the voltage across primary and secondary windings, i_1 and i_2 be the respective currents, and let n_1 and n_2 be the respective number of turns. Following relations hold for an ideal transformer

$$\frac{v_1}{n_1} = \frac{v_2}{n_2}, \tag{2.1}$$

and

$$n_1 i_1 = n_2 i_2. \tag{2.2}$$

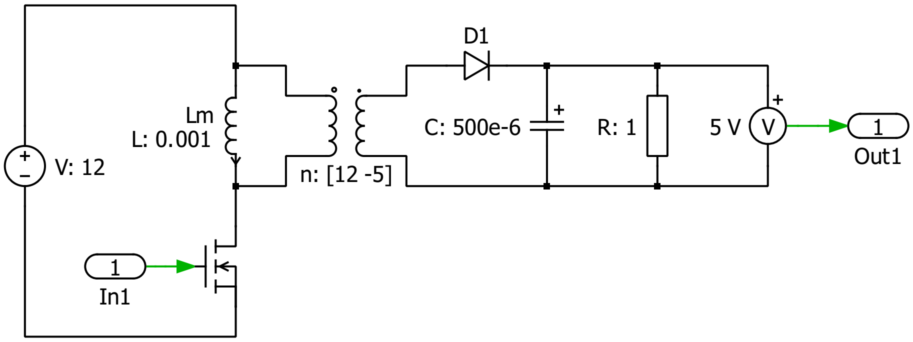


Figure 2.1: Schematic diagram of the flyback converter.

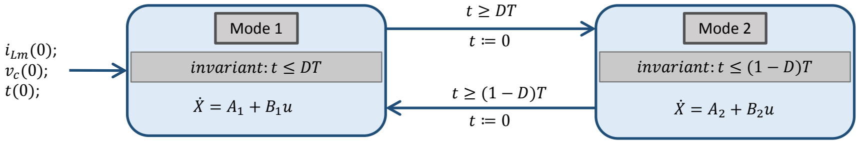


Figure 2.2: Hybrid automaton model for flyback converter.

2.1 Flyback Converter Modeling

We consider the flyback converter in open-loop configuration as shown in Figure 2.1 exported from PLECS software [9], a power electronics circuit simulator. The switching is realized by the MOSFET switch and the diode D_1. The state variables are defined by the voltage across the capacitor v_C, and current through the magnetizing inductor inductor i_{Lm}. The MOSFET switch is operated by a pulse generator of constant duty cycle D, over the switching time period T. The operation of this circuit is dependent upon the state of the MOSFET switch, i.e., being ON and OFF, resulting into two modes:

1. Mode 1: In this mode, the MOSFET switch is ON during the switching cycle $0 < t \leq DT$, wherein, the input DC voltage V_{in} is connected to the primary of the transformer. This induces the current in the secondary winding in opposite polarity to reverse bias the diode (setting it to OFF state). In this mode, the primary of the transformer is charged, wheres, the diode acts as an open switch causing the capacitor to discharge through the load resistance. We model the MOSFET switching loss by a series resistor r_{sw}. The ordinary differential equations (ODEs) for i_{Lm} and v_C for this mode are formed using conventional Kirchoff' voltage law (KVL) and Kirchoff's current law (KCL). Applying KVL on the left loop gives

$$\frac{di_{Lm}}{dt} = \frac{r_{sw}}{L_m} i_{Lm} + \frac{V_{in}}{L_m}, \tag{2.3}$$

whereas, applying KVL on the loop containing R and C gives

$$\frac{dv_C}{dt} = \frac{1}{RC} v_C. \tag{2.4}$$

[1] The tool is available online from the SpaceEx website at: http://spaceex.imag.fr/.

[2] The executable models are included on the ARCH website and are also available online from the HyST website at: http://verivital.com/hyst/.

The state space matrices, during the switching cycle $0 < t \leq DT$, are thus given by

$$A_1 = \begin{bmatrix} \frac{r_{sw}}{L_m} & 0 \\ 0 & \frac{1}{RC} \end{bmatrix}, B_1 = \begin{bmatrix} \frac{1}{L_m} \\ 0 \end{bmatrix}, X = \begin{bmatrix} i_{Lm} \\ v_C \end{bmatrix}, u = V_{in}, \tag{2.5}$$

2. Mode 2: In this mode, the MOSFET switch is OFF during the switching cycle $DT < t \leq T$, thus the input DC power supply is disconnected from the primary of the transformer. The current in the secondary flows in upward direction hence diode is forward biased (in ON state). We first consider the primary winding loop and apply KVL. Using Equation 2.1, the voltage across the primary is given by

$$v_1 = -nv_C, \tag{2.6}$$

such that the negative sign is due to its opposite direction. Applying KVL in the primary winding loop, we obtain following relation for the magnetizing inductor current

$$\frac{di_{Lm}}{dt} = -\frac{n}{L_m} v_C. \tag{2.7}$$

The current through primary winding is the same as current through L_m. From Equation 2.2, the current through the secondary winding is given by

$$i_2 = ni_{Lm}. \tag{2.8}$$

Consider the node joining R and C. The current entering this node is i_2. Applying KCL on this node, we get

$$\frac{dv_C}{dt} = \frac{n}{C} i_{Lm} - \frac{1}{RC} v_C. \tag{2.9}$$

The corresponding state space matrices, during the switching cycle $DT < t \leq T$, are thus given by

$$A_2 = \begin{bmatrix} 0 & -\frac{n}{L_m} \\ \frac{n}{C} & -\frac{1}{RC} \end{bmatrix}, B_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{2.10}$$

We have formulated a hybrid automaton model of flyback converter using the above ODEs as shown in Figure 2.2. The component values used in the model are mentioned in Figure 2.1, and adopted from [9].

2.2 Forward Converter Modeling

The forward converter may be regarded as a transformer-isolated buck converter, as illustrated in Figure 2.3 sketched using PLECS [9]. It has a MOSFET switch, and three diodes D_1, D_2, and D_3 to realize the switching operation. We consider three state variables, i.e, magnetizing current i_{Lm}, inductor current i_L, and capacitor voltage v_C. Let n_1, n_2, and n_3 be the number of turns in three windings of the transformer. The switching modes depend on the state of the MOSFET switch as well as the fact that whether inductor current $i_L \leq 0$ and the magnetizing current $i_{Lm} \leq 0$. This results in six different modes as under.

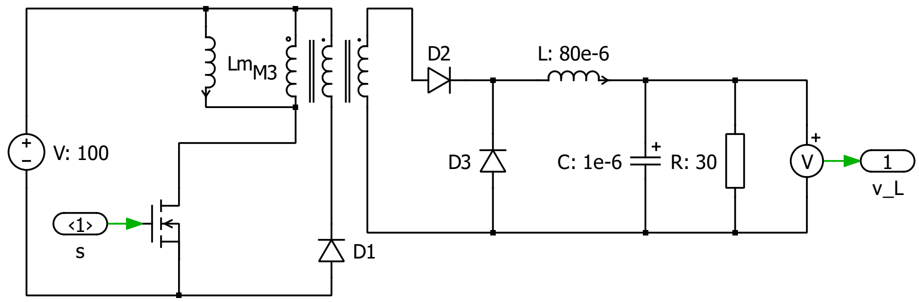


Figure 2.3: Schematic diagram of forward converter.

1. Mode 1: In this mode, the MOSFET switch is ON during the switching cycle $0 < t \leq DT$, wherein, the input DC voltage V_{in} is connected to the primary winding of the transformer. This causes D_2 to become forward biased (ON), and D_1 and D_3 to become reverse biased (OFF). Applying KVL to left most loop results in

$$\frac{di_{Lm}}{dt} = \frac{V_{in}}{L_m}, \tag{2.11}$$

whereas, the voltage across D_3 is $\frac{n_3}{n_1}V_{in}$. Applying KVL to the loop containing L and C, results

$$\frac{di_L}{dt} = \frac{n_3}{n_1 L}V_{in} - \frac{1}{L}v_C. \tag{2.12}$$

Consider the node common to L, C, and R. Applying KCL here results

$$\frac{dv_C}{dt} = \frac{1}{C}i_L - \frac{1}{RC}v_C. \tag{2.13}$$

The corresponding state space matrices, during the switching cycle $0 < t \leq DT$, are thus given by

$$A_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{L} \\ 0 & \frac{1}{C} & -\frac{1}{RC} \end{bmatrix}, B_1 = \begin{bmatrix} \frac{1}{L_m} \\ \frac{n_3}{n_1 L} \\ 0 \end{bmatrix}, X = \begin{bmatrix} i_{Lm} \\ i_L \\ v_C \end{bmatrix}, u = V_{in}. \tag{2.14}$$

2. Mode 2: The MOSFET switch is OFF during the switching cycle $DT < t \leq (1-D)T$ such that V_{in} is disconnected from the primary winding, and both $i_{Lm} > 0$ and $i_L > 0$. The diodes D_1 and D_3 are ON, whereas, D_2 is OFF. The input voltage is applied to the winding 2 of the transformer such that the voltage across L_m is $-V_{in}\frac{n_1}{n_2}$. This results in decrease of i_{Lm} such that

$$\frac{di_{Lm}}{dt} = -\frac{n_1 V_{in}}{n_2 L_m}, \tag{2.15}$$

Since L discharges through the load resistor, D_3 remains ON, such that V_{in} is not available to charge the inductor L. This gives us

$$\frac{di_L}{dt} = -\frac{1}{L}v_C, \ \frac{dv_C}{dt} = \frac{1}{C}i_L - \frac{1}{RC}v_C. \tag{2.16}$$

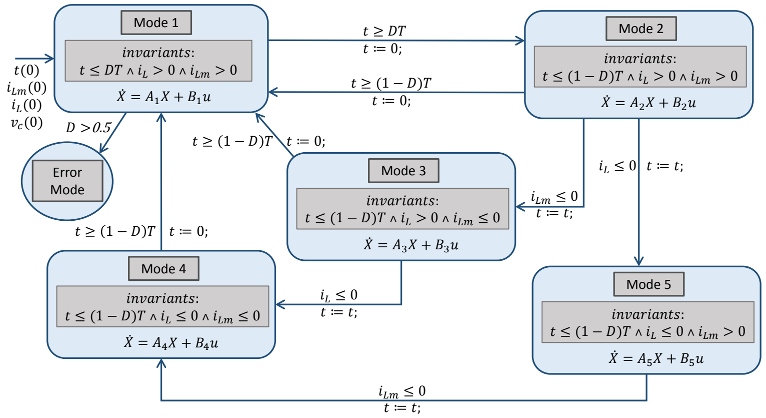


Figure 2.4: Hybrid automaton model for forward converter.

The corresponding state space matrices are

$$A_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{L} \\ 0 & \frac{1}{C} & -\frac{1}{RC} \end{bmatrix}, B_2 = \begin{bmatrix} -\frac{n_1}{n_2 L_m} \\ 0 \\ 0 \end{bmatrix}. \tag{2.17}$$

3. Mode 3: The MOSFET switch is still OFF during the switching cycle $DT < t \leq (1-D)T$ such that $i_{Lm} \leq 0$ and $i_L > 0$. As $i_{Lm} \leq 0$, diode D_1 becomes OFF. Overall, the MOSFET switch and diodes D_1 and D_2 are OFF. We can form another set of ODEs as

$$\frac{di_{Lm}}{dt} = 0, \ \frac{di_L}{dt} = -\frac{1}{L}v_C, \ \frac{dv_C}{dt} = \frac{1}{C}i_L - \frac{1}{RC}v_C. \tag{2.18}$$

The corresponding state space matrices are

$$A_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{L} \\ 0 & \frac{1}{C} & -\frac{1}{RC} \end{bmatrix}, B_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \tag{2.19}$$

4. Mode 4: The MOSFET switch is OFF during the switching cycle $DT < t \leq (1-D)T$ such that both and $i_{Lm} \leq 0$ and $i_L \leq 0$. Following set of ODEs can be formed

$$\frac{di_{Lm}}{dt} = 0, \ \frac{di_L}{dt} = 0, \ \frac{dv_C}{dt} = -\frac{1}{RC}v_C. \tag{2.20}$$

The corresponding state space matrices are

$$A_4 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{RC} \end{bmatrix}, B_4 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \tag{2.21}$$

5. Mode 5: The MOSFET switch is OFF during the switching cycle $DT < t \leq (1-D)T$. There is another possibility that i_L approaches zero while i_{Lm} is still non-zero, thus we have another condition $i_{Lm} > 0$ and $i_L \leq 0$. This gives us

$$\frac{di_{Lm}}{dt} = -\frac{n_1 V_{in}}{n_2 L_m}, \ \frac{di_L}{dt} = 0, \ \frac{dv_C}{dt} = -\frac{1}{RC} v_C. \tag{2.22}$$

The corresponding state space matrices are

$$A_5 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{RC} \end{bmatrix}, B_5 = \begin{bmatrix} -\frac{n_1}{n_2 L_m} \\ 0 \\ 0 \end{bmatrix}. \tag{2.23}$$

6. Error Mode: Inherently, the maximum possible duty cycle for the forward converter is $D \leq 0.5$. Accordingly, we have added the error mode in the model to accommodate any deadlocks due to wrong selection of parameters.

Using the above ODEs and modes, the hybrid automaton model of forward converter is formulated and shown in Figure 2.4. The component values used in the model are mentioned in Figure 2.3 and adopted from [10].

2.3 Closed-loop Forward Converter

We have also modeled the forward converter in closed-loop configuration and typically used the hysteresis control methodology as outlined in [3]. In this control methodology, the capacitor voltage v_C is allowed to vary within a hysteresis band. The hysteresis band is formed by defining an upper switching boundary, $V_{ref} + \Delta$, and a lower switching boundary, $V_{ref} - \Delta$, where V_{ref} is the desired output voltage, and Δ is the tolerance level. The state space description of the model remains the same as discussed in Section 2.3 and shown in Figure 2.2, whereas the guards $t \geq DT$ and $t \geq (1-D)T$ are changed to $v_C \geq V_{ref} + \Delta$ and $v_C \leq V_{ref} - \Delta$, respectively. Moreover, the invariants $t \leq DT$ and $t \leq (1-D)T$ are changed to $v_C \leq V_{ref} + \Delta$ and $v_C \geq V_{ref} - \Delta$, respectively.

3 SLSF Simulations and Reachability Analysis

We have automatically generated the hybrid automaton models in SpaceEx format using HyST tool and analyze these in SpaceEx environment. Moreover, we have automatically translated the same SpaceEx models into SLSF format using HyST. Formal verification of the flyback and forward converters includes verifying the corresponding capacitor voltage and inductor current to attain a stable limit cycle in settling time. For the flyback converter, we require that v_C and i_{Lm} should exhibit a stable limit within settling time t_S. For the forward converter, we require that v_C and i_L should exhibit a stable limit within settling time t_S.

SpaceEx, PLECS, and SLSF results for the capacitor voltage and inductor current are shown in Figure 3.1. It is evident from the results in Figure 3.1 that PLECS and SLSF simulation traces are contained within the over-approximated sets of reachable states. We also conclude that these results exhibit stable limit cycle, and that stable voltage is attained within 5 ms.

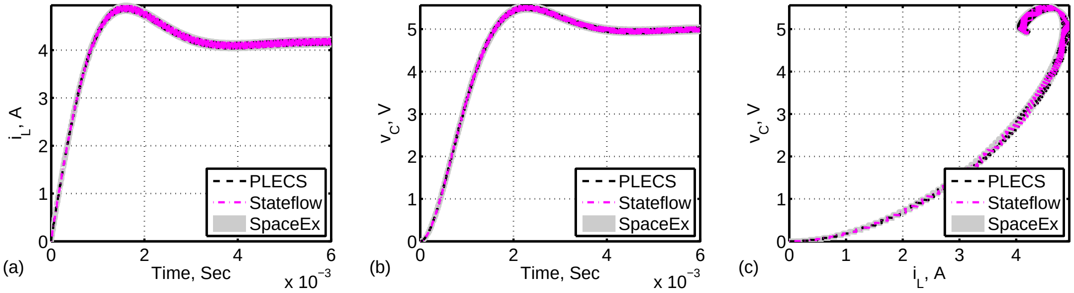


Figure 3.1: Comparison of SpaceEx reach sets, PLECS and SLSF trajectories for the flyback converter showing the simulation trace containment within overapproximated sets of reachable states: (a) Inductor current vs time (b) Capacitor voltage vs time (c) Phase-plane plot of capacitor voltage and inductor current.

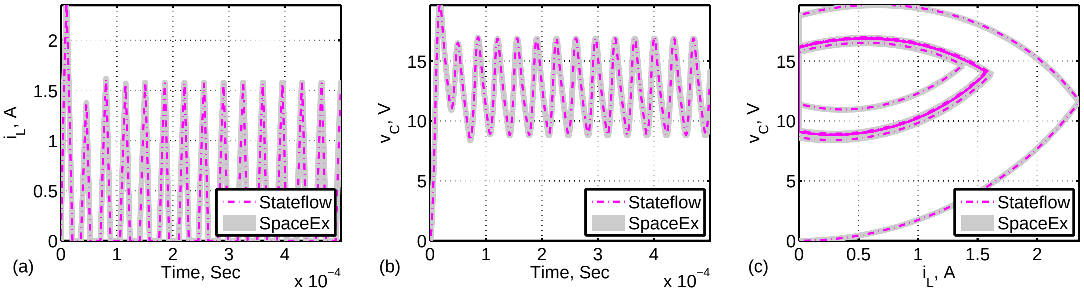


Figure 3.2: Comparison of SpaceEx overapproximations and SLSF trajectories for the open-loop forward converter, showing the simulation trace containment within overapproximated sets of reachable states: (a) Inductor current vs time (b) Capacitor voltage vs time (c) Phase-plane plot of capacitor voltage and inductor current.

We perform the reachability analysis using SpaceEx for forward converter as shown in Figure 3.2. The SLSF time traces are contained within the over-approximated sets of reachable states computed using SpaceEx. We also conclude that these results exhibit a stable limit cycle within 100 μs.

There are various sources of non-determinism in both the models such as the input voltage (V_{in}), initialization values of various state variables, the duty cycle of the PWM signal (D), and the time period of PWM signals (T). We have modeled the non-determinism of these parameters for both types of converters.

3.1 Reachability Analysis Results - Non-Determinism in Flyback Converter

First we consider the non-determinism in V_{in} for the flyback converter, such that it is allowed to vary from $11.9 - 12.1\ V$. The reachability analysis results are computed using SpaceEx and shown in Figure 3.3. We consider the variations in initial values of all the states variables, i.e., i_{Lm} and v_C. The state variable i_{Lm} is initialized for a range of $0 - 0.5\ A$, whereas v_C is initialized for $0 - 0.5\ V$. The reachability analysis results are computed using SpaceEx and shown in Figure 3.4. Next we consider non-determinism in D, such that it is allowed to vary

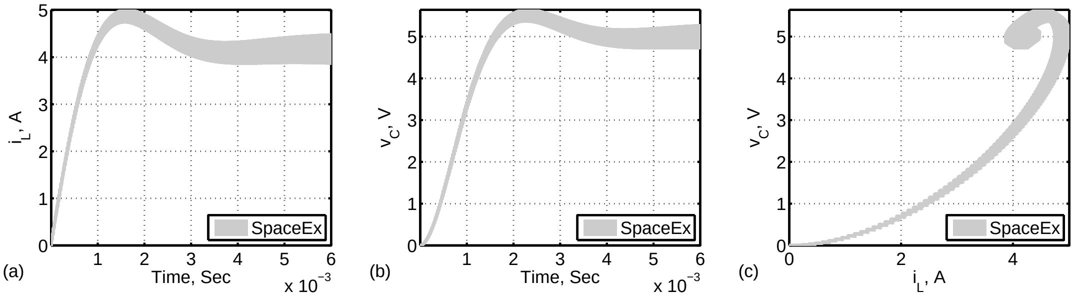


Figure 3.3: For the flyback converter model, we cater the non-determinism for the input voltage V_{in} and overapproximations are computed using SpaceEx: (a) Inductor current vs time (b) Capacitor voltage vs time (c) Phase-plane plot of capacitor voltage and inductor current.

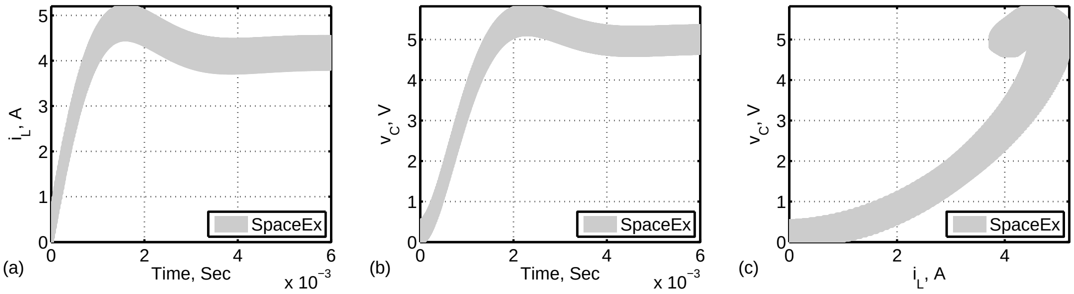


Figure 3.4: For the flyback converter model, we cater the non-determinism in initial values of i_{Lm} and v_C and overapproximations are computed using SpaceEx: (a) Inductor current vs time (b) Capacitor voltage vs time (c) Phase-plane plot of capacitor voltage and inductor current.

from $0.449 - 0.501$ s. The overapproximations computed using SpaceEx are shown in Figure 3.5. In the last, we consider the variations in T and obtain the reachability analysis results using SpaceEx as T varies between $19.96 - 20.04$ μs, as shown in Figure 3.6.

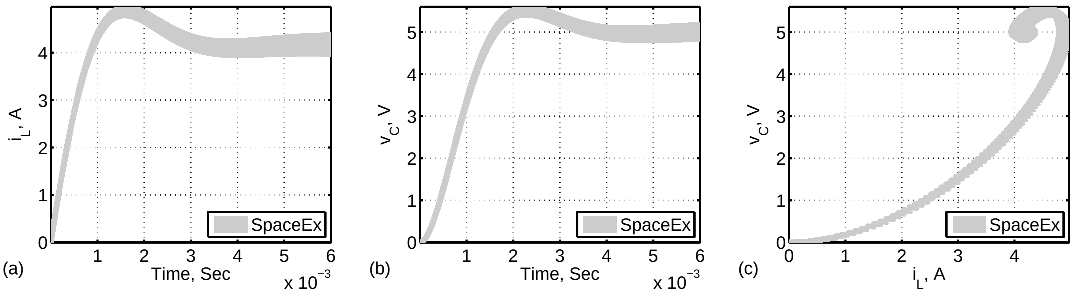


Figure 3.5: For the flyback converter model, we cater the non-determinism in the duty cycle D and overapproximations are computed using SpaceEx: (a) Inductor current vs time (b) Capacitor voltage vs time (c) Phase-plane plot of capacitor voltage and inductor current.

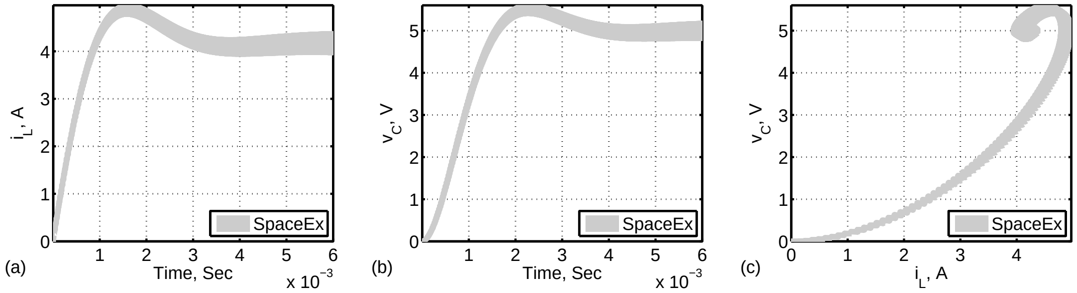


Figure 3.6: For the flyback converter model, we cater the non-determinism in the sampling time T and overapproximations are computed using SpaceEx: (a) Inductor current vs time (b) Capacitor voltage vs time (c) Phase-plane plot of capacitor voltage and inductor current.

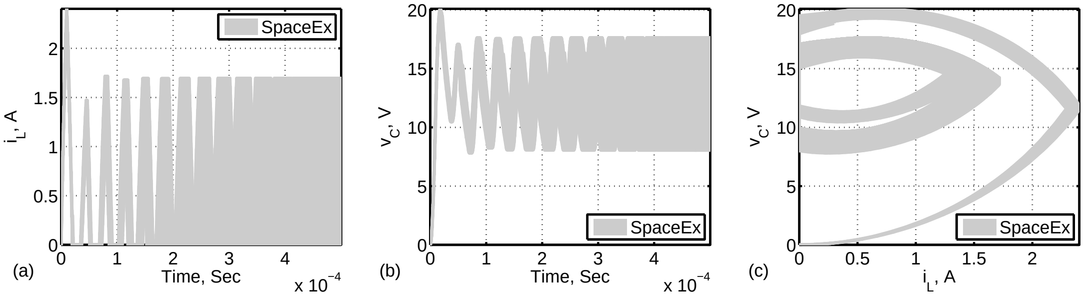


Figure 3.7: For the forward converter model, we cater the non-determinism for the input voltage V_{in} and overapproximations are computed using SpaceEx: (a) Inductor current vs time (b) Capacitor voltage vs time (c) Phase-plane plot of capacitor voltage and inductor current.

3.2 Reachability Analysis Results - Non-Determinism in Forward Converter

We consider the non-determinism in V_{in} for the forward converter, such that it is allowed to vary from $98-102\ V$. The reachability analysis results are computed using SpaceEx and shown in Figure 3.7. We model the variations in initial values of all the states variables, i.e., i_{Lm}, i_L, and v_C. The state variables i_{Lm} and i_L are both initialized for a range of $0-0.4\ A$, and v_C is initialized for $0-0.4\ V$. The reachability analysis results are computed using SpaceEx and shown in Figure 3.8. Next we consider non-determinism in D, such that it is allowed to vary from $0.39-0.41\ s$. The overapproximations computed using SpaceEx are shown in Figure 3.9. In the last, we consider the variations in T and obtain the reachability analysis results using SpaceEx as T varies between $24.39-25.64\ \mu s$, as shown in Figure 3.10.

3.3 Reachability Analysis Results - Closed-loop Forward Converter

In the last part, we present the reachability analysis results for the closed-loop forward converter using hysteresis control in Figure 3.11. For the hystersis-controlled forward converter we require that i_L and v_C should exhibit a stable limit cycle within the settling time t_S. As evident in Figure 3.11, both i_L and v_C exhibit a stable limit cycle within 50 μs.

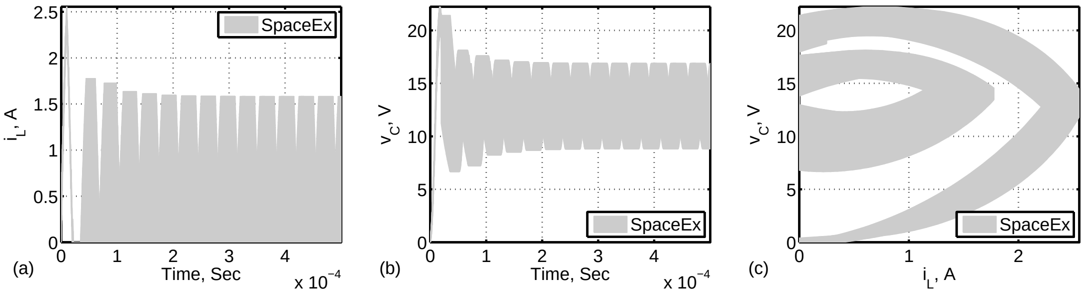


Figure 3.8: For the forward converter model, we cater the non-determinism in initial values of i_L, i_{Lm} and v_C and overapproximations are computed using SpaceEx: (a) Inductor current vs time (b) Capacitor voltage vs time (c) Phase-plane plot of capacitor voltage and inductor current.

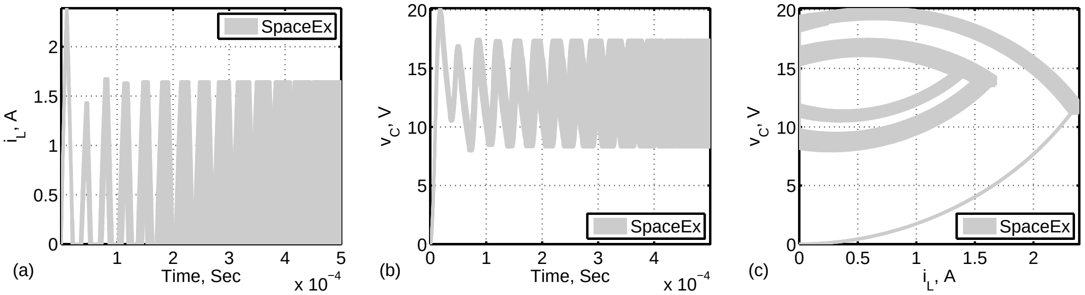


Figure 3.9: For the forward converter model, we cater the non-determinism in the duty cycle D and overapproximations are computed using SpaceEx: (a) Inductor current vs time (b) Capacitor voltage vs time (c) Phase-plane plot of capacitor voltage and inductor current.

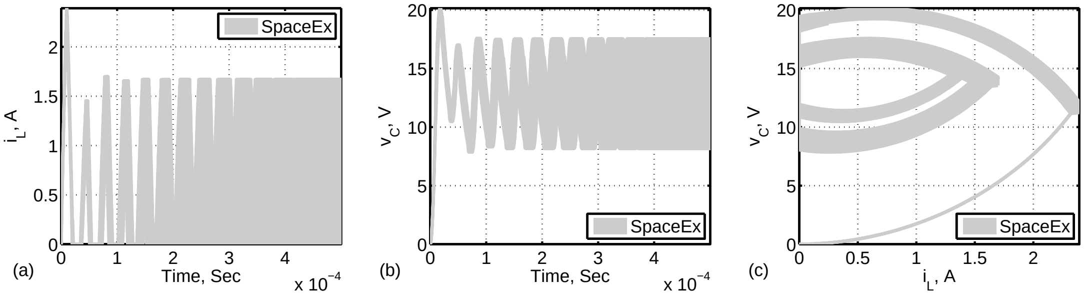


Figure 3.10: For the forward converter model, we cater the non-determinism in the sampling time T and overapproximations are computed using SpaceEx: (a) Inductor current vs time (b) Capacitor voltage vs time (c) Phase-plane plot of capacitor voltage and inductor current.

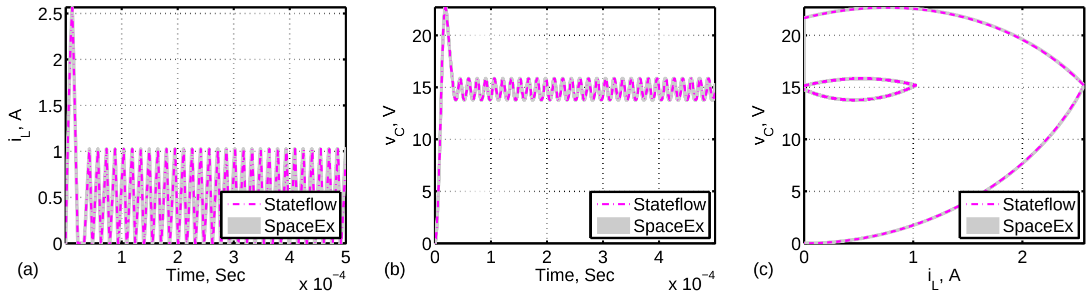


Figure 3.11: Comparison of SpaceEx and SLSF results for the hysteresis-controlled forward converter, showing the simulation trace containment within overapproximated sets of reachable states: (a) Inductor current vs time (b) Capacitor voltage vs time (c) Phase-plane plot of capacitor voltage and inductor current.

4 Key Observations

Hybrid automaton modeling and reachability analysis of transformer-isolated flyback converter has medium difficulty level. However, modeling and analysis of forward converter is more complex with three state variables and five modes. We have only used SpaceEx to perform the reachability analysis. In addition other reachability analysis tools may also be used for the reachability analysis.

We have not considered the parasitics in modeling of transformer-isolated DC-DC converters that will further increase the difficulty level of this benchmark.

5 Benchmark Outlook

On the whole, these verification benchmarks can serve as a first step towards a benchmark library to evaluate reachability and verification methods for various types of DC-DC converters. These benchmarks are open to the continuous and hybrid systems verification community to evaluate their methods and tools.

Acknowledgments The material presented in this paper is based upon the work supported by the National Science Foundation (NSF) under grant numbers ECCS 1137354, CNS 1464311, and CCF 1527398, the Air Force Research Laboratory (AFRL) through contract number FA8750-15-1-0105, and the Air Force Office of Scientific Research (AFOSR) under contract number FA9550-15-1-0258. The U.S. government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of AFRL, AFOSR, or NSF.

The authors would also like to thank the anonymous reviewers whose valuable suggestions have added value to these benchmarks and enabled us to improve upon the results.

References

[1] Matthias Althoff, Akshay Rajhans, Bruce H. Krogh, Soner Yaldiz, Xin Li, and Larry Pileggi. Formal verification of phase-locked loops using reachability analysis and continuization. *Commun. ACM*, 56(10):97–104, October 2013.

[2] Stanley Bak, Sergiy Bogomolov, and Taylor T Johnson. Hyst: a source transformation and translation tool for hybrid automaton models. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, pages 128–133. ACM, 2015.

[3] Omar A. Beg, Houssam Abbas, Taylor T. Johnson, and Ali Davoudi. Model validation of pwm dc-dc converters. *IEEE Transactions on Industrial Electronics*, 2017. DOI: 10.1109/TIE.2017.2688961.

[4] Dae-Kyu Choi et al. A novel power conversion circuit for cost-effective battery-fuel cell hybrid systems. *Journal of Power Sources*, 152:245 – 255, 2005.

[5] Goran Frehse et al. Spaceex: Scalable verification of hybrid systems. In Shaz Qadeer Ganesh Gopalakrishnan, editor, *Proc. 23rd International Conference on Computer Aided Verification (CAV)*, LNCS. Springer, 2011.

[6] Shamina Hossain, Sairaj Dhople, and Taylor T. Johnson. Reachability analysis of closed-loop switching power converters. In *Power and Energy Conference at Illinois (PECI)*, pages 130–134, 2013.

[7] Taylor T. Johnson, Zhihao Hong, and A. Kapoor. Design verification methods for switching power converters. In *Power and Energy Conference at Illinois (PECI), 2012 IEEE*, pages 1–6, February 2012.

[8] Luan Viet Nguyen and Taylor T. Johnson. Benchmark: Dc-to-dc switched-mode power converters (buck converters, boost converters, and buck-boost converters). In *Applied Verification for Continuous and Hybrid Systems Workshop (ARCH 2014)*, Berlin, Germany, April 2014.

[9] Plexim Inc., Cambridge, MA, USA. *PLECS Manual Version 4.0.4*, 2016.

[10] A. Taut et al. Educational matlab tool for simulating of forward converters. In *Proceedings of International Spring Seminar on Electronics Technology*, pages 641–644, May 2011.

[11] Toyota. Defect information report (nhtsa recall 14v-053), Feb. 12 2014.

[12] Toyota. Defect information report (nhtsa recall 15v-449), Jul. 15 2015.

Computer-Aided Formal Verification of Power Electronics Circuits

Omar Ali Beg[*], Luan Nguyen[†], Ali Davoudi[*] and Taylor T. Johnson[‡]
[*]Department of Electrical Engineering
University of Texas at Arlington, Arlington, Texas 76019, USA
Email: omar.beg@mavs.uta.edu and davoudi@uta.edu
[†]Department of Computer Science and Engineering
University of Texas at Arlington, Arlington, Texas 76019, USA
Email: luanvnguyen@mavs.uta.edu
[‡]Department of Electrical Engineering and Computer Science
Vanderbilt University, Nashville, Tennessee 37235, USA
Email: taylor.johnson@vanderbilt.edu

***Abstract*—Formal verification requires extensive analysis of a given mathematical model with respect to some correctness requirements using various tools and techniques. Manually constructing models of a given device in various formats requires considerable time and efforts. Thus we automatically generate the hybrid automaton models in SpaceEx format using HyST (Hybrid Source Transformer) tool, which is a source-to-source transformation and translation tool. We then automatically translate these SpaceEx models into Mathworks Simulink Stateflow (SLSF) for analysis thus saving significant amount of time and efforts. We present various power electronics circuits benchmarks to demonstrate the efficiency and effectiveness of HyST in model-based design process. Safe and reliable operation of these circuits in safety-critical applications necessitates a rigorous modeling and verification process. In this work, we use SpaceEx reachability analysis tool for formal verification of such circuits. We have used this computer-aided modeling technique to automatically generate and translate the models and verify that the output of a given model remains within a defined stable region in steady state.**

I. Introduction

Formal verification involves constructing a mathematical model $\mathcal{M}$ with precise semantics, extensive analysis with respect to some correctness requirement $\mathcal{P}$, and verifying that $\mathcal{M} \models \mathcal{P}$ [1]. Reachability analysis has been used for formal verification of pre-defined correctness requirements for analog mixed signal circuits [2]. In this work, we use SpaceEx [3], a reachability analysis tool, for formal verification of power electronics circuits[1]. Since one needs to build the model of a given device in various formats so as to perform extenive analysis using various tools for formal verification. Manually building the models in various formats requires significant time and efforts. Therefore, we have used a new tool HyST (Hybrid Source Transformer) [4] to automatically generate the hybrid automaton models in SpaceEx compatible format. We also use HyST to automatically convert the hybrid automaton models developed in SpaceEx to MathWorks Simulink/Stateflow (SLSF) models [2]. It is a source-to-source transformation and translation tool that takes input in the SpaceEx model format, and translates it to various other formats such as HyCreate, Flow*, dReach, C2E2, Passel 2.0, and HyComp. HyST tool is being updated over the time to add support for other analysis tools. The verification and validation research community is encouraged to use HyST as this computer-automated analysis saves significant time and efforts in model-based design process.

Power electronics form the energy middle-ware and used in automobiles, industrial automation, aerospace, and defense. Power electronics devices, such as DC-DC power converters contain switching components which lead to discrete behaviors, and have passive components that exhibit continuous dynamics within each discrete event. Such devices can be modeled as hybrid automata to perform reachability analysis. A signifcant rise in the safety recalls of cars manufactured by automotive industry due to malfunction of power electronics devices has been reported. As an exmaple, about 700,000 Toyota Prius cars were recalled in year 2014 due to an error in interaction between a boost converter and its software controller [5]. Later in year 2015, more than 100,000 Toyota Prius cars were recalled due to an inverter malfunction [6]. Therefore, such mission-critical devices would require formal verification prior implementation.

In this paper, we demonstrate effectiveness of HyST tool in automatic model-based design and formal verification process using four case studies of power electronics circuits. First two being special types of DC-DC power converters called center-tapped Buck and boost converters. In the last two case studies, we use two improved models of the transformer-isolated DC-DC power converters that were earlier presented in [7], namely, flyback converter (that acts as a Buck-boost converter) and forward converter (that acts as a Buck converter). This work is continuation of a series of benchmarks for power

[1] The tool is available online from the SpaceEx website at: http://spaceex.imag.fr/.

[2] The executable models are available online from the HyST website at: http://verivital.com/hyst/.

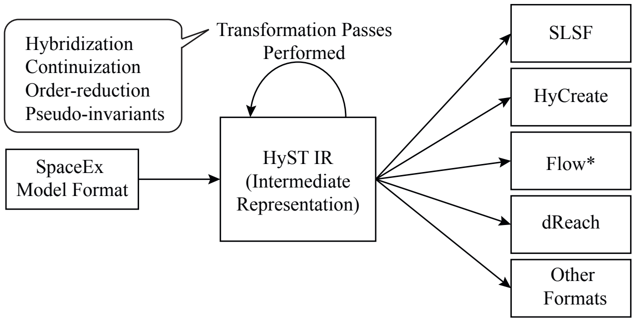


Fig. 1. Overview of the HyST conversion process.

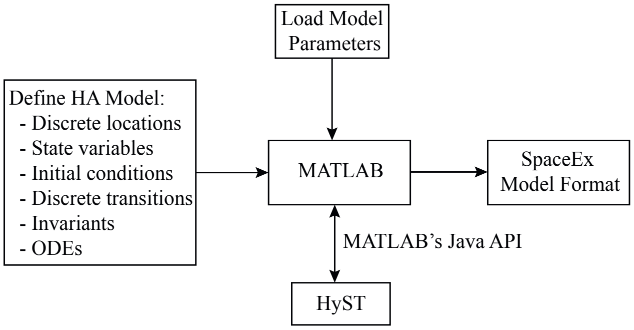


Fig. 2. Overview of automatic model generation in SpaceEx format.

electronics circuits [8]–[10] that are being developed to benefit from formal verification prior to field implementation and deployment.

II. Automatic Model Generation Using HyST

HyST is an automatic source-to-source model translation and transformation tool that takes input in SpaceEx format and generates models in SLSF, HyCreate, Flow*, dReach, C2E2, Passel, and HyComp formats [4]. The support for other reachability analysis tools will be added from time to time. HyST can be beneficial to the hybrid systems verification community in following ways:

1. The user may automatically generate a model file for numerous other tools, carry out the analysis, and choose the best suitable tool for the system under consideration.
2. The researcher involved in development of hybrid systems model checkers may quickly compare the performance of the newly developed tool with other tools.

HyST takes input in SpaceEx source format, parses it into an intermediate representation (IR), and finally prints the output source in a format specified by the user. This conversion architecture is shown in Fig. 1. IR is implemented as Java data structures to encode the hybrid automaton model components, whereas, transformation passes may be regarded as the model-to-model conversions. More details regarding HyST can be found in [4].

In this paper, we use HyST as a benchmark generator for automatic generation of hybrid automata models in SpaceEx format. Thus the user needs not to manually create the hybrid automata models through SpaceEx model editor saving considerable time and effort. We use MATLAB's API (application program interface) for Java that enables MATLAB to interact with Java programs synchronously or asynchronously. In this automatic model generation process, we need to instantiate the model components per Definition 2.1.

Definition 2.1: We define a *hybrid automaton model* by a tuple $\mathcal{M} = \langle L, X, Init, \mathcal{T}, Inv, F\rangle$, where:

- $L = \{l_1, l_2, ..., l_N\}$ is a finite set of discrete locations.
- X is a finite set of continuous state variables, such that $\forall\ x \in X\ \exists\ val(x) \in \mathbb{R}$, where $val(x)$ is a vluation of x resulted due to function mapping.
- $Init \subseteq L_0 \times X_0$ is a set of initial conditions, such that $L_0 \subseteq L$ and $X_0 \subseteq X$.
- $\mathcal{T} = \langle l_s, l_e, g, r\rangle$ is a set of feasible discrete transitions allowed among the discrete locations, where the corresponding elements of the tuple are the start location, end location, relevant guard, and the subsequent reset, respectively.
- Inv is a finite set of invariants for each discrete location.
- F is a set of ordinary differential equations (ODEs) that are defined for each location $l \in L$ over the continuous variables $x \in X$.

We implement following steps (Fig. 2) to automatically generate the hybrid automaton model using MATLAB:

1. Instantiate the matrix/string to define various components of the hybrid automaton model as per Definition 2.1.
2. Load parameter values and initialize the state variables.
3. Call the parser in HyST to represent these components into SpaceEx data structures.
4. Print into the SpaceEx model format, i.e., '.cfg', and '.xml' files.
5. Translate and print the model into the SLSF format.

III. Hybrid Automaton Model Formulation

The power electronics devices can be modeled as hybrid automata as these exhibit both the continuous and discrete behaviors due to the inherent passive elements and switches, respectivley [11]. In this section, we discuss the modeling of such circuits for use in automatic SpaceEx model generation process and translation to SLSF format. We demonstrate the effectiveness of HyST tool in model-based design process using four different types of power electronics circuits.

For the model formulation, we assume the transformer losses to be negligible. The winding at the input is called primary, whereas that towards the output is called secondary. The dynamics of such circuits depends on the operation of the MOSFET switch, i.e., being ON and OFF. We consider open loop DC-DC power converters such that the MOSFET switch is operated by a pulse generator of constant duty cycle D, over the switching time period T. The state variables are defined by the voltage across the capacitor v_C, and current through the inductor i_L.

A. Center-Tapped Buck Converter Model

It is a special type of DC-DC Buck converter, wherein, the inductor is center-tapped, i.e., a contact is made to a point

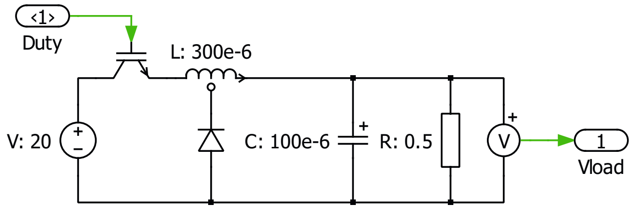


Fig. 3. Schematic diagram of center-tapped Buck converter.

halfway along the winding of an inductor. The schematic of the converter is shown in Fig. 3. Let n be the turns ratio of primary to secondary windings, n_1 be the number of turns before the center-tap, and n_2 after the center-tap. For a tapped inductor, let v_L be the overall voltage across the entire number of turns, then

$$n = \frac{v_L}{v_2} = \frac{n_1 + n_2}{n_2} = 1 + \frac{n_1}{n_2}. \tag{1}$$

The state of the MOSFET switch, i.e., being ON and OFF, results into two modes. The third mode results, when the MOSFET switch is OFF and $i_L \leq 0$.

1. Mode 1: During the switching cycle $0 < t \leq DT$, MOSFET switch is ON and diode is OFF. The input DC voltage source V_{in} supplies the primary of the inductor. In this mode, the entire inductor is charged and diodes acts as an open switch to charge the capacitor and supply the load resistance. The ODEs for i_L and v_C may be formulated using conventional Kirchoff' voltage law (KVL) and Kirchoff's current law (KCL). We use KVL on the outer loop containing L, R, and C that results in

$$\frac{di_L}{dt} = -\frac{1}{L}v_C + \frac{V_{in}}{L}, \tag{2}$$

whereas, applying KCL on the node joining L, R, and C results

$$\frac{dv_C}{dt} = \frac{1}{C}i_L - \frac{1}{RC}v_C. \tag{3}$$

The state space matrices, during the switching cycle $0 < t \leq DT$, are thus given by

$$A_1 = \begin{bmatrix} 0 & -\frac{1}{L} \\ \frac{1}{C} & -\frac{1}{RC} \end{bmatrix}, B_1 = \begin{bmatrix} \frac{1}{L} \\ 0 \end{bmatrix}, X = \begin{bmatrix} i_L \\ v_C \end{bmatrix}, u = V_{in}. \tag{4}$$

2. Mode 2: In this mode, the MOSFET switch is OFF during the switching cycle $DT < t \leq T$, thus V_{in} is disconnected from the primary of the transformer. However, the current in the secondary (equivalent to ni_L as derived from (1)) still flows hence the diode is forward biased (in ON state). We first consider the secondary winding loop, apply KVL and use (1) to form ODE as

$$\frac{di_L}{dt} = -\frac{n}{L}v_C. \tag{5}$$

Applying KCL on the node joining L, R, and C, we obtain following ODE.

$$\frac{dv_C}{dt} = -\frac{n}{C}i_L - \frac{1}{RC}v_C. \tag{6}$$

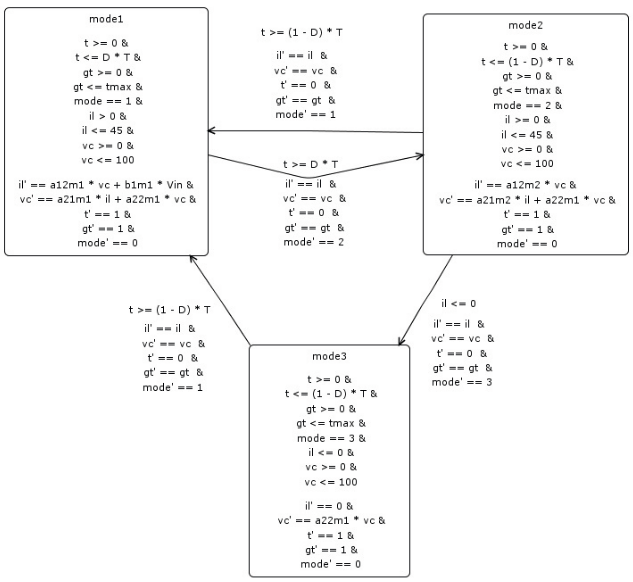


Fig. 4. Hybrid automaton model in SpaceEx format is automatically generated using HyST for center-tapped Buck converter.

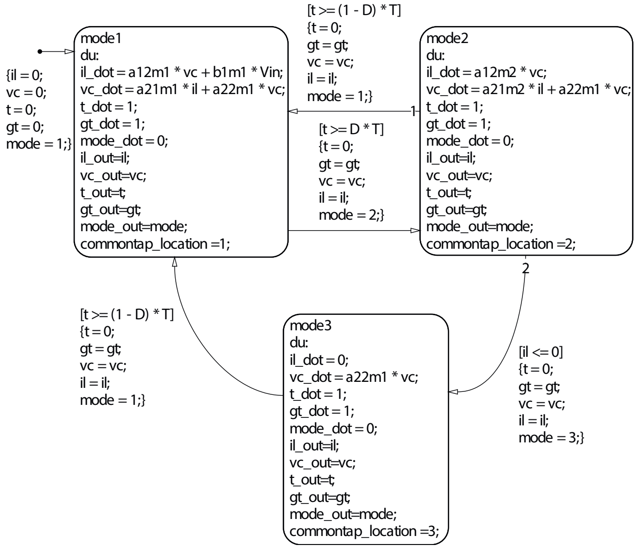


Fig. 5. SLSF model is automatically generated using HyST for center-tapped Buck converter.

The corresponding state space matrices, during the switching cycle $DT < t \leq T$, are thus given by

$$A_2 = \begin{bmatrix} 0 & -\frac{n}{L} \\ \frac{n}{C} & -\frac{1}{RC} \end{bmatrix}, B_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \tag{7}$$

We skip the ODEs for the third mode being quite straightforward. Using HyST, we have automatically generated the models of Buck converter based on above ODEs in SpaceEx and SLSF formats as shown in Fig. 4 and Fig. 5, respectively. The component values used in the model are mentioned in Fig. 3, and adopted from [12].

B. Center-Tapped Boost Converter Model

It is a special type of DC-DC boost converter with a center-tapped inductor as shown in Fig. 6. As in the above

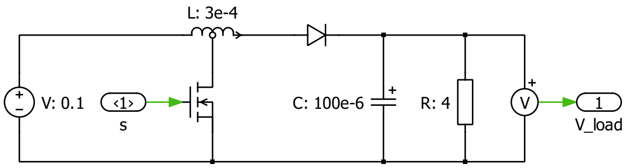


Fig. 6. Schematic diagram of the center-tapped boost converter.

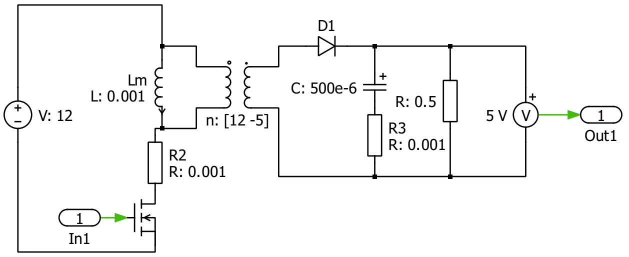


Fig. 7. Schematic diagram of flyback converter.

case, the dynamics of the circuit depends on the operation of the MOSFET switch resulting in two modes. We have automatically generated the models of center-tapped boost converter in SpaceEx and SLSF formats using HyST. Due to space limitation, we skip the formulation of ODEs and corresponding model figures. The component values used in the model are mentioned in Fig. 6.

C. Improved Model of Flyback Converter

For flyback and forward transformer-isolated DC-DC power converters, we model the transformer by L_m, a parallel magnetizing inductance, at the input side. The magnetizing current through L_m is denoted by i_{Lm}. In case of the flyback converter there are two state variables (i.e., i_{Lm} and v_C) and two modes. A simple model was presented in [7] for this type of transformer-isolated converter. This model may be improved by adding an ESR (equivalent series resistor) for the capacitor [13] as shown in Fig. 7. For space limitation, we skip the detailed model formulation. We have automatically generated SpaceEx and SLSF models of flyback converter as shown in Fig. 8 and Fig. 9, respectively. The component values used in the model are mentioned in Fig. 7, and adopted from [12].

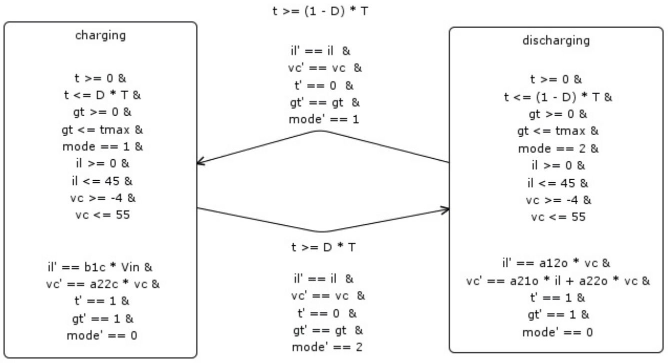


Fig. 8. Hybrid automaton model in SpaceEx format is automatically generated using HyST for flyback converter.

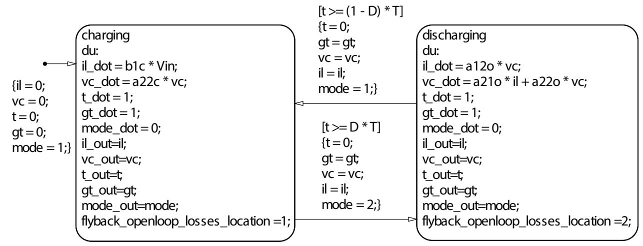


Fig. 9. SLSF model is automatically generated using HyST for flyback converter.

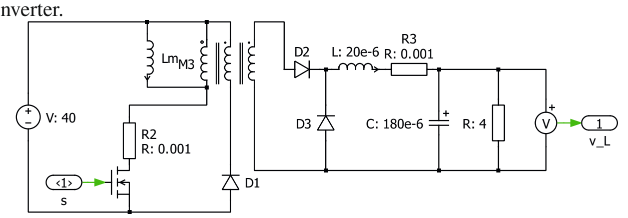


Fig. 10. Schematic diagram of forward converter.

D. Improved Model of Forward Converter

We present an improved model of the forward converter that was earlier presented in [7] to include the MOSFET switching loss (modeled by a series resistance r_{sw}) and ESR (r_L) for the inductor, as illustrated in Fig. 10. There are three state variables, i.e., i_{Lm}, i_L, and v_C. The switching modes depend on the state of the MOSFET switch as well as the fact that whether $i_L \leq 0$ and $i_{Lm} \leq 0$. This results in five different modes as shown in Fig. 11 and Fig. 12 for SpaceEx and SLSF models, respectively. Due to space limitation, we skip the ODE formulation. The component values used in the model are mentioned in Fig. 10.

E. Formal Requirements for Verification of Power Electronics Circuits

Formal verification requires that a given model of a power electronics device does not violate a predefined stability specification. We use the Lyapunov stability to define this

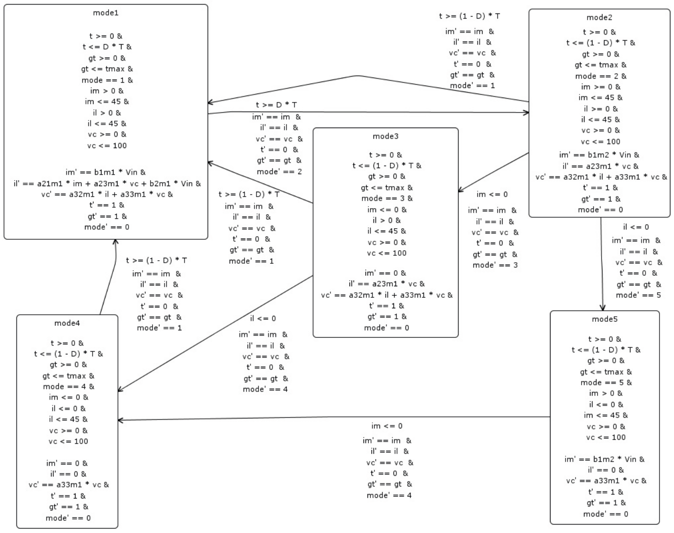
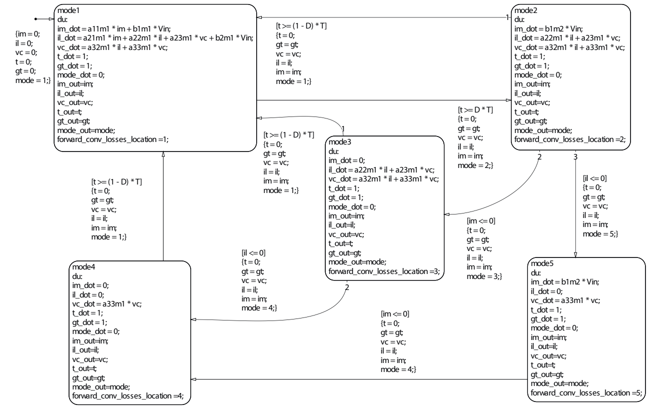


Fig. 11. Hybrid automaton model in SpaceEx format is automatically generated using HyST for forward converter.



```
{im = 0;
il = 0;
vc = 0;
t = 0;
gt = 0;
mode = 1;}
```

```
mode1
du:
im_dot = a11m1 * im + b1m1 * Vin;
il_dot = a21m1 * im + a22m1 * il + a23m1 * vc + b2m1 * Vin;
vc_dot = a32m1 * il + a33m1 * vc;
t_dot = 1;
gt_dot = 1;
mode_dot = 0;
im_out=im;
il_out=il;
vc_out=vc;
t_out=t;
gt_out=gt;
mode_out=mode;
forward_conv_losses_location =1;
```

```
[t >= (1 - D) * T]
{t = 0;
gt = gt;
vc = vc;
il = il;
im = im;
mode = 1;}
```

```
mode2
du:
im_dot = b1m2 * Vin;
il_dot = a22m1 * il + a23m1 * vc;
vc_dot = a32m1 * il + a33m1 * vc;
t_dot = 1;
gt_dot = 1;
mode_dot = 0;
im_out=im;
il_out=il;
vc_out=vc;
t_out=t;
gt_out=gt;
mode_out=mode;
forward_conv_losses_location =2;
```

```
[t >= D * T]
{t = 0;
gt = gt;
vc = vc;
il = il;
im = im;
mode = 2;}
```

```
[t >= (1 - D) * T]
{t = 0;
gt = gt;
vc = vc;
il = il;
im = im;
mode = 1;}
```

```
mode3
du:
im_dot = 0;
il_dot = a22m1 * il + a23m1 * vc;
vc_dot = a32m1 * il + a33m1 * vc;
t_dot = 1;
gt_dot = 1;
mode_dot = 0;
im_out=im;
il_out=il;
vc_out=vc;
t_out=t;
gt_out=gt;
mode_out=mode;
forward_conv_losses_location =3;
```

```
[t >= (1 - D) * T]
{t = 0;
gt = gt;
vc = vc;
il = il;
im = im;
mode = 1;}
```

```
[im <= 0]
{t = 0;
gt = gt;
vc = vc;
il = il;
im = im;
mode = 3;}
```

```
[il <= 0]
{t = 0;
gt = gt;
vc = vc;
il = il;
im = im;
mode = 5;}
```

```
[il <= 0]
{t = 0;
gt = gt;
vc = vc;
il = il;
im = im;
mode = 4;}
```

```
mode4
du:
im_dot = 0;
il_dot = 0;
vc_dot = a33m1 * vc;
t_dot = 1;
gt_dot = 1;
mode_dot = 0;
im_out=im;
il_out=il;
vc_out=vc;
t_out=t;
gt_out=gt;
mode_out=mode;
forward_conv_losses_location =4;
```

```
[im <= 0]
{t = 0;
gt = gt;
vc = vc;
il = il;
im = im;
mode = 4;}
```

```
mode5
du:
im_dot = b1m2 * Vin;
il_dot = 0;
vc_dot = a33m1 * vc;
t_dot = 1;
gt_dot = 1;
mode_dot = 0;
im_out=im;
il_out=il;
vc_out=vc;
t_out=t;
gt_out=gt;
mode_out=mode;
forward_conv_losses_location =5;
```

Fig. 12. SLSF model is automatically generated using HyST for forward converter.

specification, i.e., $\dot{x} = f(x(t))$ is stable if $\forall\ \epsilon > 0\ \exists\ \delta > 0$ such that if $\|x(0)\| \leq \delta \Rightarrow \|x(t)\| \leq \epsilon\ \forall\ t \geq 0$. Therefore, we may define a bounded region and verify that the output of the power electronics device eventually reaches and always remains in this stable region. This is hypothetically equivalent to requiring that both the state variables of interest, i.e., i_L and v_C attain a stable limit cycle in finite time. Accordingly, we define the stability specification for DC-DC power converters in steady state, such that i_L and v_C should attain a stable limit cycle within a finite settling time t_S.

IV. SLSF Simulations and Reachability Analysis

We have automatically generated SpaceEx models using HyST tool and analyze these in SpaceEx environment. We have also automatically translated the same SpaceEx models into SLSF format using HyST. For the flyback converter, we require that v_C and i_{Lm} should exhibit a stable limit within settling time t_S. For the center-tapped Buck, boost, and forward converters, we require that v_C and i_L should exhibit a stable limit within settling time t_S.

For center-tapped Buck, center-tapped boost, flyback, and forward converters, the SpaceEx and SLSF results for the capacitor voltage and inductor current are shown in Fig. 13, Fig. 14, Fig. 15, and Fig. 16, respectively. SLSF simulation traces are contained within the over-approximated sets of reachable states computed using SpaceEx. We also conclude that these results exhibit stable limit cycle, and that stable voltage is attained within 1.5 ms, 5 ms, 3 ms, and 2 ms for the respective power converters.

V. Conclusion

HyST significantly reduces the time and efforts in model-based design process and formal verification. Verification and validation research community may use HyST to automatically transform the hybrid automaton models in SpaceEx format to other formats and perform reachability analysis using aforesaid model checking tools. The hybrid automaton models of power electronics circuits that we provide in this paper form part of a benchmark library. It is being developed to evaluate various reachability analysis and verification methods. This benchmark library is open to the continuous and hybrid systems verification community for testing and evaluation of their methods and tools.

Acknowledgments

The material presented in this paper is based upon work supported by the National Science Foundation (NSF) under grant numbers CNS 1464311, ECCN 1405173, and SHF 1527398, the Air Force Research Laboratory (AFRL) through contract numbers FA8750-15-1-0105 and FA8650-12-3-7255 via subcontract number WBSC 7255 SOI VU 0001, and the Air Force Office of Scientific Research (AFOSR) under contract numbers FA9550-15-1-0258 and FA9550-16-1-0246. The U.S. government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of AFRL, AFOSR, or NSF.

References

[1] R. Alur, "Formal verification of hybrid systems," in *Embedded Software (EMSOFT), 2011 Proceedings of the International Conference on*. IEEE, 2011, pp. 273–278.

[2] M. Althoff, A. Rajhans, B. H. Krogh, S. Yaldiz, X. Li, and L. Pileggi, "Formal verification of phase-locked loops using reachability analysis and continuization," *Commun. ACM*, vol. 56, no. 10, pp. 97–104, Oct. 2013. [Online]. Available: http://doi.acm.org/10.1145/2507771.2507783

[3] G. Frehse *et al.*, "Spaceex: Scalable verification of hybrid systems," in *Proc. 23rd International Conference on Computer Aided Verification (CAV)*, ser. LNCS, S. Q. Ganesh Gopalakrishnan, Ed. Springer, 2011.

[4] S. Bak, S. Bogomolov, and T. T. Johnson, "Hyst: a source transformation and translation tool for hybrid automaton models," in *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*. ACM, 2015, pp. 128–133.

[5] Toyota. (2014, Feb. 12) Defect information report (nhtsa recall 14v-053). [Online]. Available: http://www-odi.nhtsa.dot.gov/acms/cs/jaxrs/download/doc/UCM450071/RCDNN-14V053-0945.pdf

[6] ——. (2015, Jul. 15) Defect information report (nhtsa recall 15v-449). [Online]. Available: http://www-odi.nhtsa.dot.gov/acms/cs/jaxrs/download/doc/UCM482439/RCORRD-15V449-4622.pdf

[7] O. A. Beg, A. Davoudi, and T. T. Johnson, "Reachability analysis of transformer-isolated dc-dc converters," in *Proceedings of Applied Verification for Continuous and Hybrid Systems Workshop (ARCH)*, Pittsburgh,PA, Apr. 2017, pp. 1–13.

[8] T. T. Johnson, Z. Hong, and A. Kapoor, "Design verification methods for switching power converters," in *Proceedings of the 3rd IEEE Power and Energy Conference at Illinois (PECI)*, Urbana, Illinois, USA, Feb. 2012, pp. 1–6.

[9] S. Hossain, S. Dhople, and T. T. Johnson, "Reachability analysis of closed-loop switching power converters," in *Proceedings of the 4th IEEE Power and Energy Conference at Illinois (PECI)*, Urbana, Illinois, USA, Feb. 2013.

[10] L. V. Nguyen and T. T. Johnson, "Benchmark: Dc-to-dc switched-mode power converters (buck converters, boost converters, and buck-boost converters)," in *Applied Verification for Continuous and Hybrid Systems Workshop (ARCH)*, Berlin, Germany, Apr. 2014.

[11] O. A. Beg, H. Abbas, T. T. Johnson, and A. Davoudi, "Model validation of pwm dc-dc converters," *IEEE Transactions on Industrial Electronics*, 2017, dOI: 10.1109/TIE.2017.2688961.

[12] *PLECS Manual Version 4.0.4*, Plexim Inc., Cambridge, MA, USA, 2016.

[13] S. K. Pandey, S. Patil, and V. S. Rajguru, "Isolated flyback converter designing modeling and suitable control strategies," in *Proceedings of the International Conference on Advances in Power Electronics and Instrumentation Engineering*, 2014.

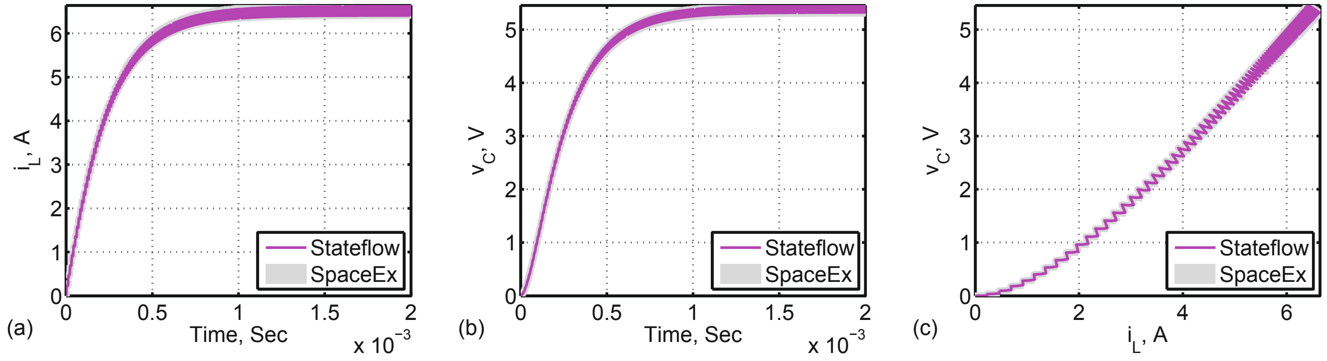


Fig. 13. Comparison of SpaceEx reach sets and SLSF trajectories for the center-tapped Buck converter showing the simulation trace containment within overapproximated sets of reachable states: (a) Inductor current vs time (b) Capacitor voltage vs time (c) Phase-plane plot of capacitor voltage and inductor current.

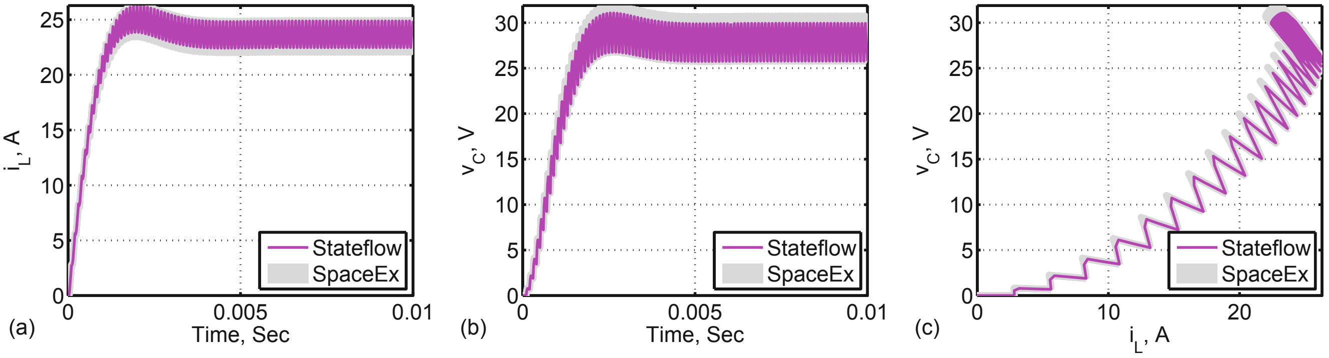


Fig. 14. Comparison of SpaceEx reach sets and SLSF trajectories for the center-tapped boost converter showing the simulation trace containment within overapproximated sets of reachable states: (a) Inductor current vs time (b) Capacitor voltage vs time (c) Phase-plane plot of capacitor voltage and inductor current.

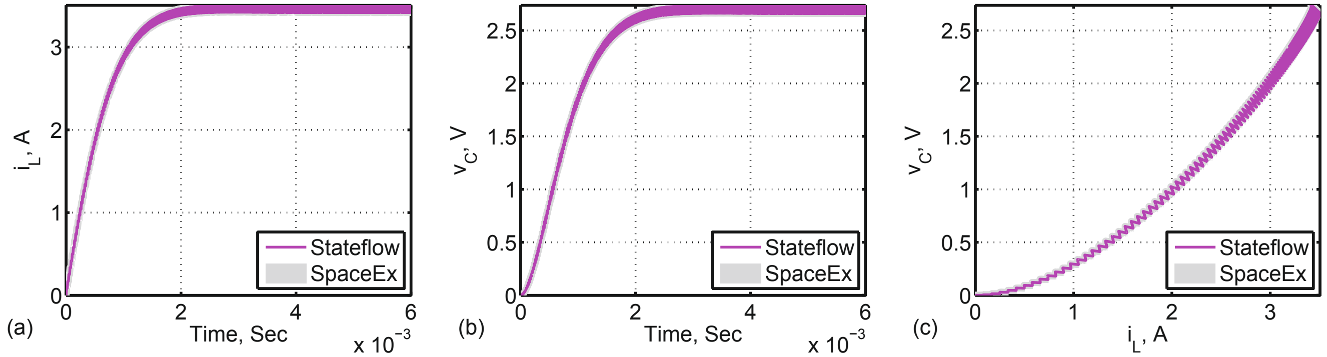


Fig. 15. Comparison of SpaceEx reach sets and SLSF trajectories for the flyback converter showing the simulation trace containment within overapproximated sets of reachable states: (a) Inductor current vs time (b) Capacitor voltage vs time (c) Phase-plane plot of capacitor voltage and inductor current.

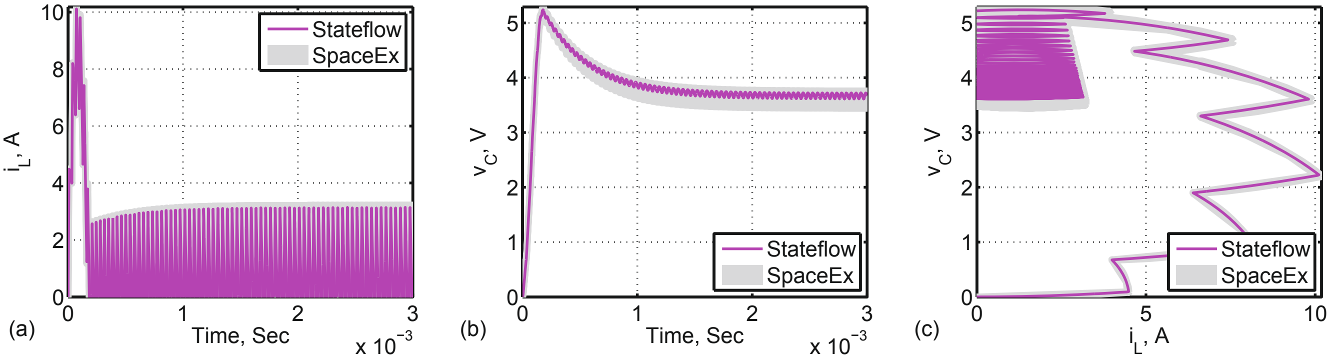


Fig. 16. Comparison of SpaceEx overapproximations and SLSF trajectories for the forward converter, showing the simulation trace containment within overapproximated sets of reachable states: (a) Inductor current vs time (b) Capacitor voltage vs time (c) Phase-plane plot of capacitor voltage and inductor current.

Model Validation of PWM DC-DC Converters

***Abstract*—This paper presents hybrid automaton modeling, comparative model validation, and formal verification of stability through reachability analysis of PWM DC-DC converters. Conformance degree provides a measure of closeness between the proposed hybrid automata models and experimental data. Non-determinism due to variations in circuit parameters is modeled using interval matrices. In direct contrast to the unsound and computationally-intensive Monte Carlo simulation, reachability analysis is introduced to overapproximate the set of reachable states and ensure stable operation of PWM DC-DC converters. Using a 200 W experimental prototype of a buck converter, hybrid automata models of open-loop and hysteresis-controlled converters are first validated against experimental data using their conformance degrees. Next, converter stability is formally verified through reachability analysis and informally validated using Monte Carlo simulations and experimental results.**

***Index Terms*—DC-DC converter, formal verification, hybrid automaton, model validation, reachability analysis.**

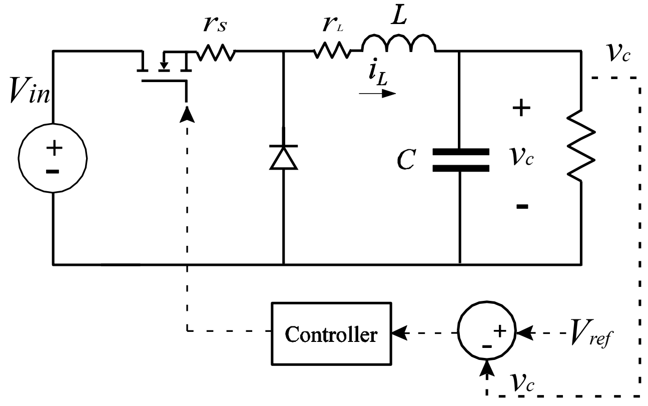


Fig. 1. Closed-loop DC-DC buck converter with main parasitic elements.

I. Introduction

ABSTRACT models of PWM DC-DC converters should reasonably match the experimental data obtained from a hardware prototype despite parametric uncertainty. More-efficient stochastic simulation techniques are based on polynomial chaos, where parametric uncertainties are accounted for by a series of orthogonal polynomials that depends upon their probability distributions [1]. Series coefficients are computed using various intrusive (e.g., stochastic Galerkin [1]) or non-intrusive (e.g., stochastic collocation [2]) methods. Examples of such stochastic methods for electrical circuits and power systems include Galerkin-based generalized polynomial chaos [3], SPICE-compatible stochastic Galerkin [4], Galerkin-based generalized decoupled polynomial chaos [5], stochastic testing [6], and SPICE-compatible stochastic collocation approach [7]. In general, polynomial chaos methods suffer from the curse of dimensionality, slow convergence with discontinuous solutions, and substantial computational overhead [8]–[11].

Another conventional approach is the simulation-based Monte Carlo paradigm [12], [13], wherein considering all possible parameter variations and initial conditions is computationally-prohibitive. Moreover, for a higher level of confidence in results produced by the Monte Carlo analysis, greater number of simulation runs are required. Generally, the total number of Monte Carlo simulations, σ, has to be increased by 100-fold to achieve additional decimal place of precision, owing to the $O(\frac{1}{\sqrt{\sigma}})$ convergence rate [14]. Conceptually, to have a full confidence in Monte Carlo results, one would require infinite number of simulation runs [15], [16]. The level of required modeling fidelity depends on the critical nature of the application domain. For example, the root cause of the 2014 recall of around 700,000 Toyota Prius cars was attributed to an error in the interaction between a boost converter and its software controller [17]. Likewise, more than 100,000 Toyota Prius cars were recalled due to an inverter failure [18]. Therefore, this mission-critical domain would require significant confidence in the modeling accuracy. At the same time, the utilized model validation tool should be conservative enough to overapproximate all possible sets of states reachable by the model execution.

The formal verification community has been using reachability analysis-based model checking tools to have sufficient confidence in the model. Therefore, we first use rigorous model validation paradigms [19] to quantify the closeness between the abstract model waveforms and experimental data using the *conformance degree* [20]. Stable converter operation is then *formally verified* on the model using reachability analysis. The boundaries of state trajectories can be found from average-value models [21], [22]. Reachability analysis overapproximates the set of all possible reachable states (i.e., the reach sets) from a given set of initial states and parameter values. One can then confidently ascertain a stable converter operation if the reach sets remain within a desired region of the state space for a given time span. Without loss of generality, we have considered a DC-DC buck converter, with main parasitic elements, as shown in Fig. 1.

General reachability analysis tools include, but are not limited to, HyTech [23], PHAVer [24], UPPAAL [25], HSolver [26], d/dt [27], Flow* [28], and SpaceEx [29]. To effectively use such model checking tools, hybrid automata models of DC-DC converters are required [30]. Hybrid automaton modeling of DC-DC converters is presented in [31]–[36]. However, [33]–[35] do not consider component losses/variations and the discontinuous conduction mode (DCM), and do not perform the reachability analysis. PHAVer in [37] computes the reach sets for an open-loop boost converter but does not include DCM or component losses. MATLAB/Ellipsoidal Toolbox is used in [38] for the reachability analysis of DC-DC converters. However, Ellipsoidal-based set computations suffer from the curse of dimensionality. SpaceEx (the successor of PHAVer) scales quite efficiently and is used as the reachability analysis tool in this paper.

We have formally defined a precise hybrid automaton model for PWM DC-DC converters, that accommodates main circuit parasitics, DCM, and the non-determinism due to parameter variations, that have not been considered altogether in any past work. We also use the notion of conformance degree to compare different model abstractions, using their output trajectories, that has not been used in any of the work cited above. Moreover, all the hybrid automata models are automatically generated. Herein, the proposed approach is shown to outperform the traditional Monte Carlo simulation in computation time. In summary, the main contributions of this paper are:

- Hybrid automata models for DC-DC converters are automatically generated, validated against Simulink/Stateflow, PLECS simulations, and hardware measurements, and verified using reachability analysis in SpaceEx. These models include component nonidealities and different operational modes.
- The conformance degree of the hybrid automata models validates these against the experimental data, by providing a proximity measure between executions/behaviors of these two in both time and space.
- Non-determinism due to parametric variations is modeled using interval matrices, which results in a set-valued additive input term in the system dynamics.
- The reachability analysis achieves a fixed point where there are no other reach sets (i.e., the model output will remain within reach sets as $t \to \infty$). It is impossible to get such success through Monte Carlo analysis.

The remainder of this paper is organized as follows: Hybrid automaton modeling is discussed in Section II. Application of conformance degree for model validation is discussed in Section III. Section IV uses interval analysis to model the non-determinism caused by the parameter variation. SpaceEx-based reachability analysis is discussed in Section V. Section VI validates the developed models against a 200 W buck converter prototype using the conformance degree, formally verifies the model properties using reachability analysis, and presents comparison with the Monte Carlo simulation. Section VII concludes the paper.

II. Hybrid Automaton Modeling

A. Preliminaries

DC-DC converters exhibit both continuous and discrete behaviors due to the presence of passive elements and switching components, respectively. Hybrid automaton modeling [39] integrates resulting differential equations and finite state machines in a single formalism. The state of a hybrid automaton model may change in two ways, i.e., through a continuous flow trajectory within a given topology (Definition 2.2), and through a discrete transition between two given topologies (Definition 2.3). A *topology* is defined as the circuit configuration in each switching sub-interval (Fig. 2). We define $\mathbb{R}^n$ as the set of n-dimensional reals, and 2^X as the power set of a given set X, i.e., the set of all the subsets of X.

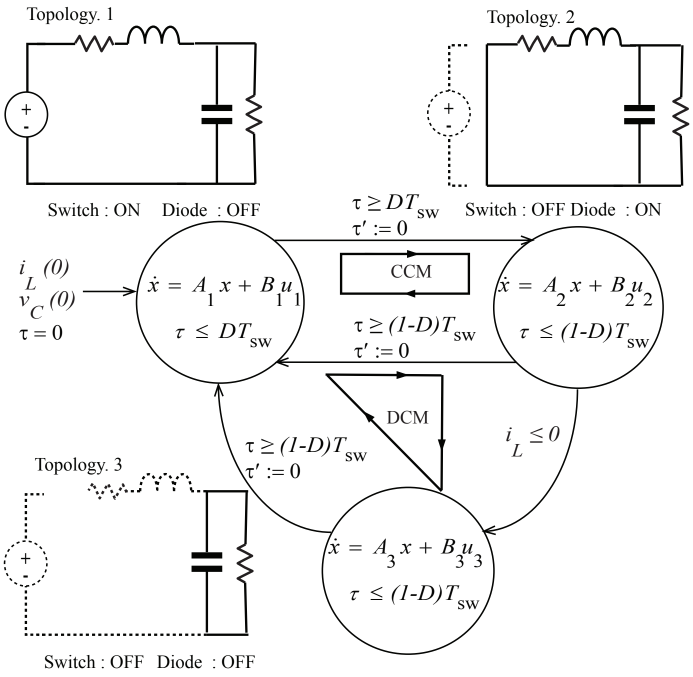


Fig. 2. Topologies, operational modes, and hybrid automaton modeling of a DC-DC buck converter.

B. Hybrid Automaton Model Syntax and Semantics

We first formally define the model components in mathematical set representation, and then define the model execution as these components interact.

Definition 2.1: A *hybrid automaton model* is defined by a tuple $\mathcal{H} = \langle Q, X, init, U, E, g, G, inv, h, F \rangle$, which has the following components:

- *Topologies:* $Q = \{q_1, q_2, ..., q_N\}$ is a finite set of topologies.
- *State Variables*: $X \subseteq \mathbb{R}^n$ is set of continuous state variables. A *state* is defined by $(q, x) \in Q \times X$.
- *Initial Conditions*: $init \subseteq Q_0 \times X_0$ is a set of initial conditions, such that $Q_0 \subseteq Q$ and $X_0 \subseteq X$.
- *Inputs*: $U = \{u_1, u_2, ...u_N\}$ is the set of inputs for each topology.
- *Discrete Transitions*: $E \subset Q \times Q$ is a set of feasible discrete transitions allowed among the topologies, such that an element $e_{ij} = (q_i, q_j) \in E$ implies that a discrete transition from i^{th} topology to j^{th} topology is allowed. It might not be possible to visit the entire set of topologies from one particular topology (Definition 2.3).
- *Guard Function*: $g : E \to G$ is the guard function that maps each element $e_{ij} \in E$ to its corresponding guard $g(e_{ij}) \in G$.
- *Guards*: $G \subseteq 2^X$ is the guard set such that $\exists\ g(e_{ij}) \in G$ for each $e_{ij} \in E$. A *guard* is a property of the hybrid automaton model that must be satisfied by a state to take a discrete transition from a given topology to another pre-defined topology. A state $(q_k, x_k) \in Q \times X$ *satisfies* $g(e_{ij})$ (i.e, $(q_k, x_k) \vDash g(e_{ij})$) iff $q_k = q_i$ and $x_k \in g(e_{ij})$.

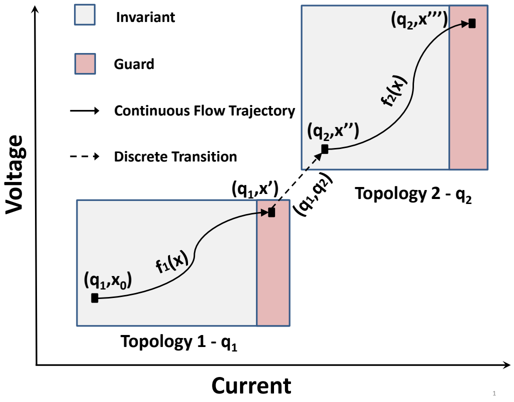


Fig. 3. Execution of the hybrid automaton model of DC-DC converters.

- *Invariants*: $inv : Q \rightarrow 2^X$ is a mapping that assigns an invariant $inv\,(q) \subseteq X$ for each topology $q \in Q$. An *invariant* is a property of the hybrid automaton model that must be satisfied by all the states for a given topology. A state $(q, x) \vDash inv(q)$ iff $x \in inv(q)$.
- *Reset of Continuous State*: $h : E \times X \rightarrow X$ resets the continuous state, i.e., if a discrete transition takes place from i^{th} topology to j^{th} topology as defined by $e_{ij} \in E$ with $x \in X$, the continuous state is reset to a new value $x' = h(e_{ij}, x) \in X$, such that $x' \in inv(q_j)$.
- *Set of ODEs*: F is the set of ordinary differential equations (ODEs) that are defined for each topology $q \in Q$ over the continuous variables $x \in X$. The continuous dynamics for each $q \in Q$ is defined by $F(q, x, u)$ over a given time horizon $t \in [\tau_1, \tau_2]$ that assigns a Lipschitz continuous vector space in $\mathbb{R}^n$.

Remarks: Here, $x' \in X$ symbolizes the new value of a continuous state $x \in X$ after a continuous flow or a discrete transition. If a state (q, x) does not satisfy an invariant $inv(q)$, then real time τ is stopped, forcing the continuous state x to stop evolving within a topology. The guard function ensures discrete transition to an appropriate topology, once the corresponding guard is satisfied. Here, invariants and guards are defined in the form of bounds over continuous state variables in Fig. 3.

Definition 2.2: The *continuous flow trajectory* $\mathcal{T}$ for a hybrid automaton model $\mathcal{H}$ is defined by the valuations of $x \in X$. For a given initial state $(q, x_0) \in Q \times X$ and $u \in U$, $\exists\, f(q, x, u) \in F$ that results in a final continuous state $x' \in X$, whereas q remains unchanged with given invariant $inv(q)$, iff $(q, x) \vDash inv(q)$. $\forall\ t \in [\tau_1\ \tau_2]$, $\mathcal{T}$ is given by

$$\mathcal{T}(q, x') = x_0 + \int_{\tau_1}^{\tau_2} f(q, x, u)dt. \tag{1}$$

and denoted by $(q, x_0) \xrightarrow{f} (q, x')$.

At each topology, converter dynamics can be modeled by ODEs; e.g., system matrices A_q and B_q describe the continuous flow trajectories in topology $q \in \{1, 2, 3\}$ of Fig. 2.

Definition 2.3: The *discrete transition* for a hybrid automaton model $\mathcal{H}$ is defined as: for a given state $(q_i, x) \in Q \times X$ and $u \in U$, there is a function $h(e_{ij}, x)$ that resets the continuous state to $x' \in X$, and the topology to q_j, iff $(q_i, x) \vDash inv(q_i)$ and $(q_i, x) \vDash g(e_{ij}) \in G$, and $\exists\ e_{ij} \in E$. The discrete transition is denoted by $(q_i, x) \xrightarrow{h} (q_j, x')$.

Definition 2.4: An *execution* of a hybrid automaton model $\mathcal{H}$ is an alternating sequence of continuous flow trajectories and discrete transitions.

The example of an execution is shown in Fig. 3.

The switching instance can be determined either externally (e.g., by a duty cycle command for the MOSFET) or internally (e.g., by meeting appropriate threshold conditions for the diode). The sequence of topologies, observed periodically in the steady state, defines an operational mode. Example of three topologies and two operational modes for a buck converter are shown in Fig. 2.

C. Model Instantiation for DC-DC Converters

We may now implement the syntax and semantics of the hybrid automaton model developed above for DC-DC converters. We define D as the duty cycle, T_{sw} as the switching period, and V_{in} as the DC input voltage. We can represent the continuous dynamics for a given topology as a standard set of state-space equations

$$\frac{dx}{dt} = A_q x + B_q u \tag{2}$$

where, $x \in \mathbb{R}^n$ is a vector of continuous states, Q is a finite set of topologies, $u \subseteq U$ such that $U \subseteq \mathbb{R}^m$ is a set of input vectors, and $A_q \in \mathbb{R}^{n \times n}$ and $B_q \in \mathbb{R}^{n \times m}$ are system matrices. Such formation can be readily created for the buck converter in Fig. 2. The instantiation of the hybrid automation model for an open-loop DC-DC converter, as per Definition 2.1, is:

- Three topologies are denoted by $Q = \{q_1, q_2, q_3\}$.
- The continuous state vector is $x = [i_L\ v_C\ \tau]'$, where τ represents real time such that $\frac{d\tau}{dt} = 1$.
- $U = \{[V_{in}, 0, 0]', [0, 0, 0]', [0, 0, 0]'\}$ forms the input vector set.
- $E = \{(q_1, q_2), (q_2, q_1), (q_2, q_3), (q_3, q_1)\}$ defines the feasible discrete transitions, e.g., (q_2, q_3) means a discrete transition from topology 2 to 3 is allowed.
- Guard set, for the corresponding elements of E, is defined by $G = \{(\tau \geq DT_{sw}), (\tau \geq (1 - D)\, T_{sw}), (i_L \leq 0), (\tau \geq (1 - D)\, T_{sw})\}$.
- The continuous flow trajectory is defined by (2), with the corresponding state matrices for each topology. For topology 1, this can be denoted by $(q_1, x_0) \xrightarrow{f} (q_1, x')$, as shown in Fig. 3. Here, (q_1, x_0) is the initial state and (q_1, x') is the final state as the automaton continuously evolves with the continuous flow dynamics $f_1(x)$.
- The reset function h defines a new continuous state x'' for the new topology. For example, if a transition is to take place from topology 1 to topology 2 with some final state $x' \in X' \subset X$ in topology 1, h assigns the new state $x'' \in X'' \subset X$ in topology 2. For topology 1 to topology 2, a discrete transition is denoted by $(q_1, x') \xrightarrow{h} (q_2, x'')$, as shown in Fig. 3.

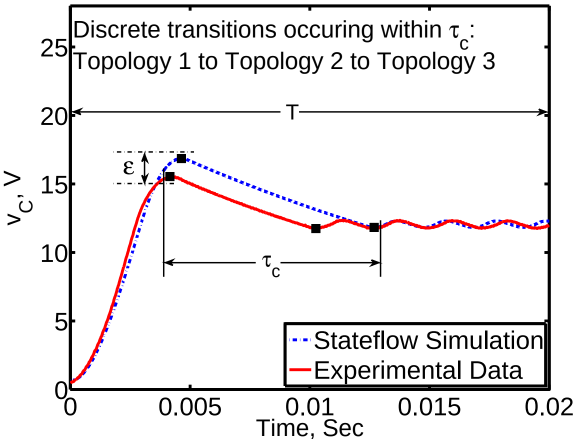


Fig. 4. Output trajectories of capacitor voltage for the closed-loop controlled buck converter - local mismatch for interval τ_c and ε.

The evolution of the hybrid automaton model starts with initial conditions from set $init$, e.g., $(q_1, x_0) \in init$ for a given input $u_1 = [V_{in}, 0, 0]'$ and, subsequently, the continuous state evolves according to the set of ODEs defined by (2) (i.e., F in Definition 2.1). The topology remains the same, i.e., $q(t) = q_1$, as x_0 evolves inside the invariant $inv(q_1)$, such that it attains a final value $x' \in inv(q_1)$. Once the continuous state x' satisfies the guard $g(e_{q_1 q_2})$ corresponding to the edge $e_{q_1 q_2} \in E$, the topology may transition from q_1 to q_2, and the continuous state is reset with a new value x'' in the new invariant set $inv(q_2) \subset X$ with a new input $u_2 = [0, 0, 0]'$.

This hybrid automaton model can be extended to closed-loop DC-DC converters, e.g., hysteresis-controlled converters. The tuple remains the same except that the guards are defined in terms of switching boundaries. The hysteresis band is formed by defining an upper switching boundary, $V_{ref} + \delta$, and a lower switching boundary, $V_{ref} - \delta$, where V_{ref} is the desired output voltage, and δ is the tolerance level. Thus, $G = \{(v_C \geq V_{ref} + \delta), (v_C \leq V_{ref} - \delta), (i_L \leq 0), (v_C \leq V_{ref} - \delta)\}$.

It should be noted that time τ does not appear in the guard expressions. Therefore, we have developed two hybrid automata models for the closed-loop buck DC-DC converter, i.e., one with variable τ (called the *time-dependent* hybrid automaton model), and another without variable τ (called the *time-independent* hybrid automaton model). For the time-independent hybrid automaton model, we perform the reachability analysis for an unbounded time, i.e., compute the reach sets as $t \to \infty$.

III. Validation through Conformance Degree

Model validation of DC-DC converters requires comparing *output trajectory* as defined by (1) for a given hybrid automaton model $\mathcal{H}$ with the measured data from an experimental prototype referred to as $\mathcal{I}$.

Our goal is to find an appropriate measure of distance for output trajectories of hybrid automata models. One can consider the output trajectories of the capacitor voltage (v_C) for a closed-loop buck converter shown in Fig. 4. The experimental data obtained from a prototype and output trajectory of the hybrid automaton model in Simulink/Stateflow are overlaid. Intuitively, the two output trajectories look similar; however, the sup norm would give a large value to the distance between them. This is, partly, because $\mathcal{I}$ and $\mathcal{H}$ might transition among various topologies at slightly different moments in time. Therefore, our distance measure should allow some *wiggle room* in time. Rather than comparing only the states that are exactly time-aligned, it should allow comparison of states that are within some $\tau_c > 0$ time units of each other.

Moreover, it is not appropriate to compare outputs when two systems have executed different numbers of discrete transitions. Thus, our distance measure must only compare states after an equal number of discrete transitions between topologies of the two systems. Note that within the time window τ_c in Fig. 4, both the hardware prototype as well as the Stateflow model exhibit two discrete transitions. To this end, we introduce the parameter $j \in \mathbb{N}$, that counts the number of discrete transitions each system makes, where $\mathbb{N}$ is the set of natural numbers. It is reasonable to require that the transition times of the two systems be close to consider that the systems themselves are close: the value τ_c will also bound the difference in transition times. The distance measure will account for the distance between output trajectories, captured by the value $\varepsilon > 0$. Thus, we have a 2-value distance measure, with values τ_c and ε capturing the time and space distance between the two output trajectories as illustrated in Fig. 4.

The output trajectories of hybrid automata models are parameterized with t and j. The time spent in a given converter topology is $t \in \mathbb{R}_{>0}$, and $j \in \mathbb{N}$ counts the number of discrete transitions between different topologies (where $\mathbb{R}_{>0}$ is the set of positive real numbers). We write $y(t, j)$ for the output trajectory at the hybrid time $(t, j) \in \mathbb{R}_{>0} \times \mathbb{N}$, i.e., at time t and after j transitions. Let $domy \subset \mathbb{R}_{>0} \times \mathbb{N}$ denote the domain of output trajectory y, i.e., the set of all (t, j), so that $(T, J, \tau_c, \varepsilon)$-closeness [20] can be formally defined.

Definition 3.1: Take an output trajectory for time $T \in \mathbb{R}_{>0}$, a maximum number of discrete transitions $J \in \mathbb{N}$, and parameters $\tau_c, \varepsilon > 0$. Two output trajectories y_1 and y_2 are $(T, J, \tau_c, \varepsilon)$-close, shown as $y_1 \approx_{(\tau_c, \varepsilon)} y_2$, if (a) for all $(t, j) \in$ domy_1 such that $t \leq T, j \leq J$, there exists $(s, j) \in$ domy_2 where $|t - s| \leq \tau_c$, and $\|y_1(t, j) - y_2(s, j)\| \leq \varepsilon$, and (b) for all $(s, j) \in$ domy_2 such that $s \leq T, j \leq J$, there exists $(t, j) \in$ domy_1 where $|t - s| \leq \tau_c$, and $\|y_2(s, j) - y_1(t, j)\| \leq \varepsilon$.

$(T, J, \tau_c, \varepsilon)$-closeness gives a proximity measure between the two output trajectories in both time and space. It shows that for every point $y_1(t, j)$, y_2 has a point $y_2(s, j)$ which is ε-close to it, and may occur anywhere in the window $[t - \tau_c, t + \tau_c]$ (and vice versa). Allowing this wiggle room in time is important when comparing the output trajectories, because the discrete transitions could occur at different times. The two values T and J limit our testing horizon. $(T, J, \tau_c, \varepsilon)$-closeness can be lifted from output trajectories to systems. One can validate the model through the conformance degree between its output trajectory and measured data.

Definition 3.2: Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be two hybrid automata models. The *conformance degree* of $\mathcal{H}_1$ to $\mathcal{H}_2$, given τ_c, is defined as the smallest ε such that for every trajectory y_1 of $\mathcal{H}_1$, there exists a trajectory y_2 of $\mathcal{H}_2$, where $y_1 \approx_{(\tau_c, \varepsilon)} y_2$. We denote this conformance degree by $\mathbf{CD}_\tau(\mathcal{H}_1, \mathcal{H}_2)$.

We will use this definition intuitively for model validation of DC-DC converters. We compute the conformance degree $\mathbf{CD}_{\tau}(\mathcal{H}_1, \mathcal{H}_2)$ for some $\tau_c > 0$ in different case studies of Section VI, and effectively say that some local mismatch is permissible within a window τ_c for the output trajectories of the models and the hardware prototype.

IV. Modeling Non-Determinism using Interval Analysis

The system matrices in the hybrid automata models of DC-DC converters depend on component values. The variations due to manufacturing tolerance, aging, and temperature result in non-determinism of component values. Analysis of electrical circuits with such variations has been reported in literature using interval arithmetic-based genetic optimization [40] and affine arithmetic [41]. We use the interval arithmetic [42] to incorporate the parameter variations within the reachability analysis framework. The range of component values are represented in terms of intervals. A real interval v is a set of real numbers given by

$$[\underline{v}, \overline{v}] = \{v \in \mathbb{R} : \underline{v} \leq v \leq \overline{v}\}, \tag{3}$$

where $\underline{v}$ is the infimum and $\overline{v}$ is the supremum. Given two intervals, $[\underline{u}, \overline{u}]$ and $[\underline{v}, \overline{v}]$, their product is another interval given by

$$[\underline{u}, \overline{u}] * [\underline{v}, \overline{v}] = [min(\underline{u}\underline{v}, \underline{u}\overline{v}, \overline{u}\underline{v}, \overline{u}\overline{v}), max(\underline{u}\underline{v}, \underline{u}\overline{v}, \overline{u}\underline{v}, \overline{u}\overline{v})]. \tag{4}$$

The quotient of two intervals, with a non-zero divisor, is

$$\frac{[\underline{u}, \overline{u}]}{[\underline{v}, \overline{v}]} = [\underline{u}, \overline{u}] * \left(\frac{1}{[\underline{v}, \overline{v}]}\right), \tag{5}$$

where

$$\left(\frac{1}{[\underline{v}, \overline{v}]}\right) = \left[\frac{1}{\overline{v}}, \frac{1}{\underline{v}}\right]. \tag{6}$$

If $[\underline{v}, \overline{v}]$ has both bounds negative, then

$$\left(\frac{1}{[\underline{v}, \overline{v}]}\right) = \left[\frac{1}{\underline{v}}, \frac{1}{\overline{v}}\right]. \tag{7}$$

These intervals may also be defined by the midpoint-radius representation

$$mid(v) = \frac{1}{2}(\underline{v} + \overline{v}), \tag{8}$$

$$rad(v) = \frac{1}{2}(\overline{v} - \underline{v}). \tag{9}$$

The interval matrix for the system matrix is $\mathcal{A} = [\underline{A}, \overline{A}]$. System stability can be deferred by examining matrix extrema, i.e., $\underline{A}$ and $\overline{A}$ [43]. Therefore, it is sufficient to consider every combination of matrix extrema to overapproximate the reach set. The overapproximation of an interval matrix $\mathcal{A}$ is given by splitting it into two parts, i.e., a nominal part and a symmetric part [44]. Consider a linear dynamic system with n state variables having single deterministic input V_{in}, with the following state-space representation

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \vdots \\ \dot{x}_n \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} V_{in}. \tag{10}$$

We use SpaceEx reachability analysis tool (discussed in Section V) to compute the reach sets for non-deterministic hybrid automaton model, with linear dynamics defined by (10) for a given topology. It may be mentioned that SpaceEx, in its present version, does not fully handle the matrix algebra operations. Hence, we need to define the state dynamics as scalar combination of other state variables. For example, for the i^{th} state variable in (10), one has

$$\dot{x}_i = a_{i1}x_1 + a_{i2}x_2 + ... + a_{ij}x_j + ... + a_{in}x_n + b_iV_{in}. \tag{11}$$

To incorporate parameter variation, one can replace the above coefficients with intervals, and write the expression as a differential inclusion

$$\dot{x}_i \in [\underline{a_{i1}}, \overline{a_{i1}}]x_1 + ...[\underline{a_{ij}}, \overline{a_{ij}}]x_i ... + [\underline{a_{in}}, \overline{a_{in}}]x_n + [\underline{b_i}, \overline{b_i}]V_{in}. \tag{12}$$

Since SpaceEx, in its present version, does not support the interval arithmetic, the intervals $[\underline{a_{ij}}, \overline{a_{ij}}]$ of (12) are computed outside the SpaceEx environment using (4), (5), (6), and (7). Subsequently, these intervals are transformed into the midpoint-radius representation to include the state and parametric intervals before implementing in the SpaceEx environment. Using (8) and (9), one can write (12) in a midpoint-radius representation as

$$\begin{aligned} \dot{x}_i \in \; & \{mid(a_{i1}) \pm rad(a_{i1})\}x_1 + ... + \{mid(a_{ij}) \pm \\ & rad(a_{ij})\}x_j + ... + \{mid(a_{in}) \pm rad(a_{in})\}x_n \\ & + \{mid(b_i) \pm rad(b_i)\}V_{in}. \end{aligned} \tag{13}$$

The mid-points correspond to the nominal parameter values that are constant terms, which can be separated as

$$\begin{aligned} \dot{x}_i \in \; & (a_{i1}x_1 + r_{i1}) + ... + (a_{ij}x_j + r_{ij}) + ... + \\ & (a_{in}x_n + r_{in}) + (b_iV_{in} + r_{bi}). \end{aligned} \tag{14}$$

This defines the continuous dynamics in the hybrid automaton model for the state variable x_i. The radii $r_{i1}, r_{i2}, ..., r_{ij}, ..., r_{in}$, and r_{bi} are expressed as product of the state and parametric intervals, such that r_{ij} is given by

$$r_{ij} \in [-rad(a_{ij}), rad(a_{ij})] * [\underline{x_j}, \overline{x_j}], \tag{15}$$

where, x_j varies between $\underline{x_j}$ and $\overline{x_j}$. For example, for the hysteresis-controlled DC-DC buck converter considered here, $\underline{v_C} = 0\ V$ and $\overline{v_C} = 20\ V$ in (15). Thus the coupling between the state variables is accommodated in the amended SpaceEx model by formulating r_{ij} in terms of $[\underline{x_j}, \overline{x_j}]$, and incorporating it in the dynamics in (14). The product of the two intervals in (15) is yet another interval, obtained using (4). The intervals thus computed are used in the model to define the lower and upper bounds for respective radii. Since this treatment of the state variables as intervals is not catered in Monte Carlo simulations, SpaceEx provides more reliable results.

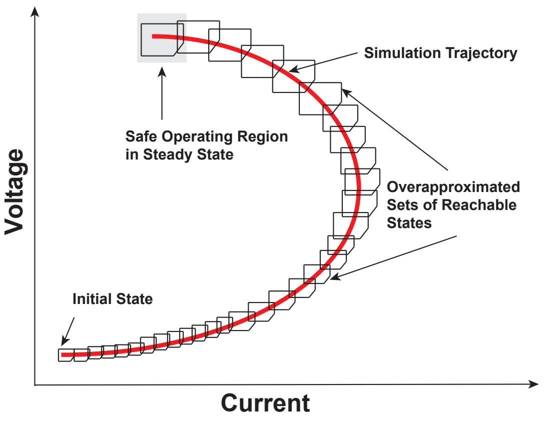

Simulation Trajectory
Safe Operating Region in Steady State
Overapproximated Sets of Reachable States
Initial State
Voltage
Current

Fig. 5. Reachability analysis using reach sets for formal verification of a hybrid automaton model.

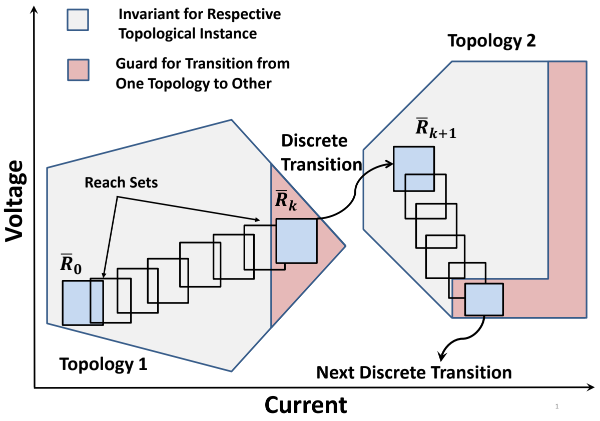


Fig. 6. Reach sets in different topologies with transitions imposed by guards.

V. Reachability Analysis for Hybrid Automata

Reachability analysis has been used by the formal verification community, and we have implemented it in its entirety for PWM DC-DC converters modeled as hybrid automata. In general, reachability analysis has been documented to produce more reliable results than Monte Carlo simulations:

1. The reachability analysis is more efficient. Monte Carlo analysis becomes computationally less tractable with increased size and complexity of a given system [38].
2. Reachability analysis is conclusive. In contrast, infinitely many Monte Carlo simulations are required to span the entire design parameter space and operational conditions and have full confidence in the final results [15], [16].
3. SpaceEx-based reachability analysis considers the entire state space [45], while Monte Carlo simulations only sample the parameter space. Generally, reachability analysis is theoretically superior and more sound [46].

We formally verify the stability properties of nondeterministic hybrid automata models of PWM DC-DC converters through the reachability analysis. We define the stability in the sense of Lyapunov, i.e., $\dot{x} = f(x(t))$ is stable if $\forall\ \theta > 0\ , \exists\ \beta > 0$ such that if $\|x(0)\| \leq \beta \Rightarrow \|x(t)\| \leq \theta\ \forall\ t \geq 0$. We may define a bounded region and verify that the output of the hybrid automaton model eventually reaches, and always remains, in this stable region, as seen in Fig. 5. We define the stability specification such that from the settling time t_s, the output voltage $V_C(t)$ should remain bounded within a tolerance γ of the reference voltage $V_{ref}(t)$, i.e., for $t \geq t_s \Rightarrow V_C(t) = V_{ref}(t) \pm \gamma$.

Definition 5.1: State x is *reachable* iff $\exists$ an execution α such that $x \in \alpha$.

Definition 5.2: The *set of reachable states* contains all the states that are reachable from a given set of initial conditions for a given time.

Consider an example of an autonomous system $\dot{x} = Ax$. The set of reachable states from initial time t_0 to final time t_f, from a given initial set X_0, is

$$\mathcal{R}_{t_0}^{t_f}(X_0) = \bigcup_{t\in[t_0,t_f]} e^{At}X_0. \tag{16}$$

However, (16) does not cater to the discrete transitions associated with the hybrid dynamical systems. Additionally, the exact set of all reachable states is undecidable [29].

In practice, overapproximations of the reachable states are computed using geometrical data structures (e.g, boxes, polytopes, ellipsoids, or zonotopes [47]), and denoted by $\overline{\mathcal{R}}$. For simplicity, we call these overapproximations as the *reach sets* in this paper. This framework can be extended to hybrid dynamical systems by including invariants and guard sets (Fig. 6), and implemented in various reachability analysis tools by software research community as mentioned in Section I. The reach sets for continuous dynamics can be computed using continuous post-operators so long as the continuous dynamics of DC-DC converter are contained within the invariant set defined for the corresponding topology or do not enter the guard set. Once the guard condition is satisfied within an invariant, a transition takes place from topology 1 to topology 2 such that the next reach set is computed using discrete post-operator. This process goes on until either the final time in a local time horizon, or a fixed point, is reached. A *fixed point* signifies that the reachability algorithm cannot find any new reach set during the current iteration other than those computed in the previous iteration. SpaceEx reachability tool computes the reach sets of a hybrid dynamical system. It is a classical fixed point algorithm based on computation of symbolic states [29].

Definition 5.3: A *symbolic state* is defined as a pair (q, Θ), where q is a topological instance, and Θ is the corresponding convex continuous set.

The reach set $\overline{\mathcal{R}}$ is obtained by computing the set of symbolic states. This reach set is the fixed point of the sequence $\overline{\mathcal{R}}_o = post_c\,(Init)$, and the successors are

$$\overline{\mathcal{R}}_{k+1} := \overline{\mathcal{R}}_k \bigcup post_c\left(post_d\left(\overline{\mathcal{R}}_k\right)\right) \tag{17}$$

where, $post_d$ is the *discrete post-operator* that defines the reach sets after a discrete transition from $\overline{\mathcal{R}}_k$. This corresponds to the h function defined in Definition 2.1. The *continuous post-operator*, $post_c$, defines the reach sets for the continuous states from $\overline{\mathcal{R}}_k$ after an arbitrary amount of time is elapsed. This corresponds to F in Definition 2.1.

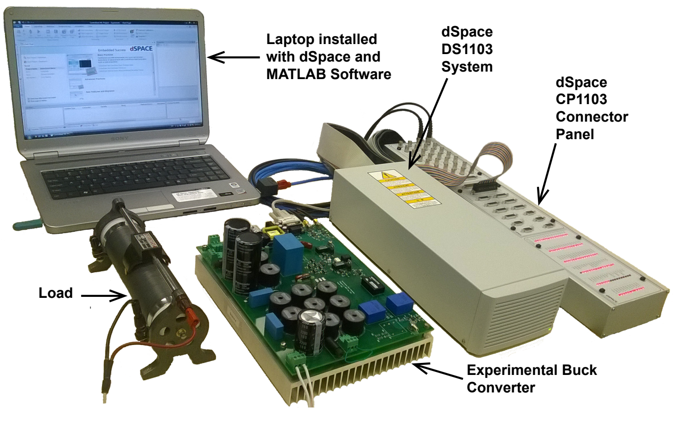

Laptop installed with dSpace and MATLAB Software
dSpace DS1103 System
dSpace CP1103 Connector Panel
Load
Experimental Buck Converter

Fig. 7. Buck converter prototype controlled with dSPACE DS1103.

An approximated computation of Θ_k is given in [29] for the k^{th} time step. Hence, a sequence of convex continuous sets $\Theta_0, \Theta_1, ...\Theta_{N-1}$ is computed to form a *flowpipe* that covers the reach sets up to a pre-defined time such that N represents the number of time steps. This flowpipe is then used to compute the transition successors. Only those states can take the transition that satisfy the guard associated with the present topology and the invariant of the target topology. This process is continued until a fixed point is reached, i.e., if all the reach sets that are computed in the present iteration, are contained in reach sets computed in the previous iteration, i.e., $\overline{\mathcal{R}}_{k+1} \subseteq \overline{\mathcal{R}}_k$. This signifies that no new reach sets could be found and the computation process may be terminated.

VI. Case Studies

An experimental setup of a buck converter, controlled with a dSpace DS1103 unit, has been prototyped, as shown in Fig. 7. The experimental results are used for benchmarking purposes against MATLAB/PLECS [48], Simulink/Stateflow [49], Monte Carlo simulations, and SpaceEx reachability analysis. Circuit parameters $L = 2.65$ mH, $C = 2.2$ mF, and $R = 10$ Ω are used throughout this study. The non-determinism due to the parameter variations is modeled using the interval matrices in SpaceEx model. For a coherent comparison in terms of parameter variations in R, L, and C, we have used 15% tolerance for Monte Carlo simulations and SpaceEx reachability analysis. We have used the Hybrid Source Transformer (HyST) which is a source-to-source conversion tool for hybrid automata models [50]. The hybrid automaton model is developed using the java interface in MATLAB, and transformed into a SpaceEx compatible model using HyST data structures. We use the conformance degree to validate the hybrid automaton model against the experimental data. Then, the reachability analysis results are provided for formal verification of an open-loop and a hysteresis-controlled buck converter.

A. Model Validation Using Conformance Degree Testing

We use notations $\mathcal{I}_O$ and $\mathcal{I}_C$ for hardware prototypes in open-loop and closed-loop configurations, respectively. PLECS and Stateflow models are denoted by $\mathcal{H}_{OP}$, $\mathcal{H}_{CP}$ and $\mathcal{H}_{OS}$, $\mathcal{H}_{CS}$, respectively, where subscript O denotes an open-loop and C denotes a closed-loop configuration. The computed ε values against τ_c (as defined in Section III) are tabulated in Table I for the corresponding output trajectories. It is evident from Table I that the ε values of $\mathcal{H}_{OP}$ and $\mathcal{H}_{OS}$ as well as $\mathcal{H}_{CP}$ and $\mathcal{H}_{CS}$ are close enough (also, as seen in Fig. 8). We have computed conformance degrees for the prototype buck converters, i.e., $\mathcal{I}_O$ and $\mathcal{I}_C$, in comparison with other models, i.e., $\mathcal{H}_{OP}$, $\mathcal{H}_{OS}$ and $\mathcal{H}_{CP}$, $\mathcal{H}_{CS}$. We also define the absolute value of the maximum difference measured between the two given output trajectories as Δ for a given time duration τ_c. The measured Δ values are tabulated in Table I. The ε values depicted in Table I provide enough wiggle room in comparison with the corresponding Δ to validate that $\mathcal{H}_{OP}$ and $\mathcal{H}_{OS}$ are reasonable abstractions for $\mathcal{I}_O$, whereas $\mathcal{H}_{CP}$ and $\mathcal{H}_{CS}$ are reasonable abstractions for $\mathcal{I}_C$. Consider, for example, the case of a closed-loop buck converter. The 1^{st} row under closed-loop configuration in Table I provides the ε value (i.e., $\varepsilon = 3.6667$ A) and Δ value (i.e., $\Delta = 3.0590$ A), as we compare the inductor current (i_L) output trajectories for PLECS and experimental prototype. Δ of the two output trajectories remain within ε (as also depicted in Fig. 9 (a)). This is also true for the corresponding output trajectories of capacitor voltage (v_C). Accordingly, the hybrid automata models are validated in conformance with both the open-loop and the closed-loop converter prototypes.

TABLE I
Conformance Degree Analysis

Config.	Type of Output Trajectories	τ_c Value (s)	ε Value	Δ Value
Open Loop	i_L - PLECS vs Experiment	3×10^{-4}	5.1515 A	4.5570 A
	i_L - Stateflow vs Experiment	3×10^{-4}	5.0008 A	4.5570 A
	i_L - Stateflow vs PLECS	3×10^{-4}	0.1785 A	0 A
	v_C - PLECS vs Experiment	3×10^{-4}	1.8945 V	1.7202 V
	v_C - Stateflow vs Experiment	3×10^{-4}	2.3201 V	1.7202 V
	v_C - Stateflow vs PLECS	3×10^{-4}	0.6666 V	0 V
Closed Loop	i_L - PLECS vs Experiment	8×10^{-4}	3.6667 A	3.0590 A
	i_L - Stateflow vs Experiment	8×10^{-4}	3.6643 A	3.0590 A
	i_L - Stateflow vs PLECS	8×10^{-4}	0.0878 A	0 A
	v_C - PLECS vs Experiment	8×10^{-4}	2.8014 V	1.5905 V
	v_C - Stateflow vs Experiment	8×10^{-4}	2.7677 V	1.5905 V
	v_C - Stateflow vs PLECS	8×10^{-4}	0.0580 V	0 V

B. Formal Verification of the Open-loop Buck Converter

We consider the voltage stability specification to perform formal verification. For example, for $t_s = 0.025$ s, and $V_{ref} = 48$ V, we define $\gamma = 6$ V. This results in an upper voltage bound of 54 V, and lower voltage bound of 42 V, as shown in Fig. 8(b) by dotted lines. The input parameters are $V_{in} = 100$ V, and $f_s = 60$ kHz. The output trajectories and phase-plane responses are considered for the startup transients of the open-loop buck converter. The parameters' variations have been modeled using interval analysis in SpaceEx model, and also included in the Monte Carlo simulation. The reachability analysis results, obtained

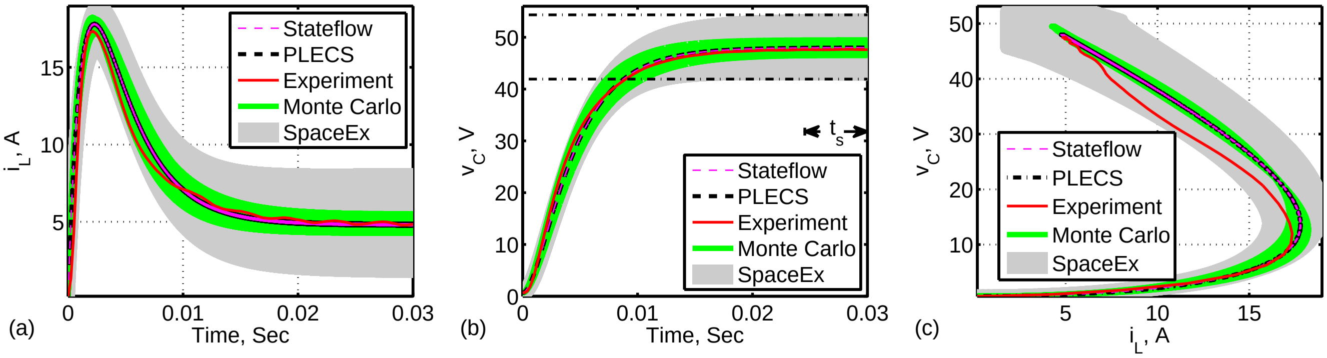


Fig. 8. Startup transients for an open-loop buck converter using interval matrices including Stateflow, PLECS, experiment, Monte Carlo, and SpaceEx; (a) current vs. time, (b) voltage vs. time, and (c) phase portrait.

using SpaceEx, are plotted in Fig. 8. It can be seen that the steady-state inductor current and capacitor voltage waveforms lie within the reachability analysis results, i.e., the simulations and measurement data are contained within the reach sets. Moreover, we verify that $v_C(t) \in [42, 54]$ for $t \geq t_s$ for Stateflow, PLECS, measurement data, Monte Carlo analysis, and SpaceEx analysis results.

C. Verification of the Hysteresis-controlled Converter

We define the voltage stability specification for the closed-loop buck converter to perform formal verification. For $t_s = 0.012$ s, and $V_{ref} = 12$ V, we define $\gamma = 1$ V. This leads to upper and lower voltage bounds of 13 and 11 V, respectively, as shown by dotted lines in Fig. 9(b). In this case study, the time-dependent and the time-independent models (as mentioned in Section II) are considered. First, SpaceEx reachability analysis is performed for the time-dependent model. The new parameters are $V_{in} = 24$ V, $V_{ref} = 12$ V, and $f_s = 50$ kHz. The trajectories are shown in Fig. 9 for Stateflow, PLECS, and experimental data along with reach sets computed using SpaceEx. The Stateflow, PLECS, and SpaceEx results match right from the start until the steady state is reached. Experimental results match that of Stateflow, PLECS, and SpaceEx in the steady state. It can be observed in Fig. 9 that Stateflow, PLECS, and measured results remain within the reach sets computed using SpaceEx, verifying $v_C(t) \in [11, 13]$ for $t \geq t_s$.

We can formally verify the time-independent SpaceEx model for an unbounded time, i.e., $t \to \infty$, by excluding τ. This would not be possible through Monte Carlo analysis as, even for a limited time span, one has to take into account infinite number of possible combinations. We have successfully achieved a fixed point using SpaceEx, with unbounded time, and with all possible parameter variations. The phase-plane plots are given for the start-up transients in Fig. 10. As seen, all results remain within the computed reach sets as $t \to \infty$, verifying $v_C(t) \in [11, 13]$ as $t \to \infty$.

A comparison of Monte Carlo analysis and SpaceEx reachability analysis, in term of computation times, is shown in Table II. Both are run on a Windows 7 SP1 (64 bit) platform, with Intel (R) core i7-2600 CPU with 3.40 GHz, 16.0 GB RAM, MATLAB version 8.5.0.197613 (R2015a), PLECS version 3.7.3, and SpaceEx version 0.9.8d. While infinite iterations are required to have full confidence in model validation through Monte Carlo analysis, we have only used finite (i.e., 2000) iterations as would be done in practice. Even then, it is evident that the SpaceEx reachability outperforms the Monte Carlo analysis in computation time, as seen in Table II.

VII. Conclusion

A hybrid automaton modeling approach for PWM DC-DC converters is developed. We have used the conformance testing for model validation when compared with a hardware prototype of DC-DC converters. The interval matrices analysis accommodates the model non-determinism caused by variations in component values. Reachability analysis frameworks are developed for formal verification of the resulting hybrid automata models. It is shown that the proposed reachability analysis outperforms the brute force Monte Carlo analysis in computation time and confidence level.

Acknowledgment

The authors would like to thank the anonymous reviewers whose valuable suggestions/comments have added value to this work and enabled us to effectively present our contributions.

References

[1] D. Xiu and G. E. Karniadakis, “The wiener–askey polynomial chaos for stochastic differential equations,” *SIAM J. Sci. Comput.*, vol. 24, no. 2, pp. 619–644, Oct 2002.

[2] D. Xiu and J. S. Hesthaven, “High-order collocation methods for differential equations with random inputs,” *SIAM J. Sci. Comput.*, vol. 27, no. 3, pp. 1118–1139, Dec 2005.

[3] A. Monti *et al.*, “A polynomial chaos theory approach to the control design of a power converter,” in *Proc. 35th Power Electron. Specialists Conf.*, Aachen, Germany, 2004, pp. 4809–4813.

[4] P. Manfredi *et al.*, “Stochastic analysis of switching power converters via deterministic spice equivalents,” *IEEE Trans. Power Electron.*, vol. 29, no. 9, pp. 4475–4478, Sept 2014.

[5] ——, “Generalized decoupled polynomial chaos for nonlinear circuits with many random parameters,” *IEEE Microw. Wireless Compon. Lett.*, vol. 25, no. 8, pp. 505–507, Aug 2015.

[6] Z. Zhang *et al.*, “Stochastic testing method for transistor-level uncertainty quantification based on generalized polynomial chaos,” *IEEE Trans. Comput.-Aided Des. Integr. Syst.*, vol. 32, no. 10, pp. 1533–1545, Oct 2013.

[7] A. Prasad and S. Roy, “Multidimensional variability analysis of complex power distribution networks via scalable stochastic collocation approach,” *IEEE Trans. Compon. Packag. Technol.*, vol. 5, no. 11, pp. 1656–1668, Nov 2015.

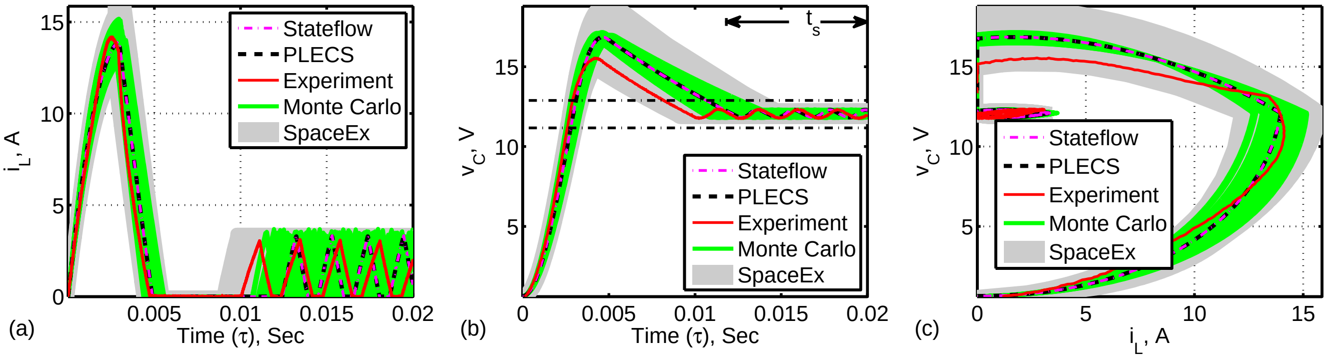


Fig. 9. Time-dependent hysteresis-controlled converter analysis using interval matrices including Stateflow, PLECS, experiment, Monte Carlo, and SpaceEx; (a) current vs. time, (b) voltage vs. time, and (c) phase portrait.

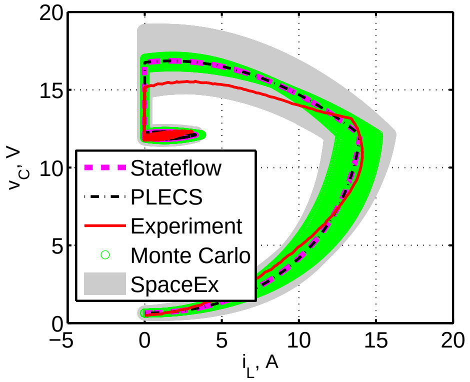


Fig. 10. Time-independent hysteresis-controlled converter analysis using interval matrices: Stateflow, PLECS, experiment, Monte Carlo, and SpaceEx.

TABLE II
COMPARISON OF MONTE CARLO AND SPACEEX ANALYSIS

System Configuration	Monte Carlo Iterations	Monte Carlo Time (s)	SpaceEx Time (s)	Times SpaceEx is Faster
Open Loop	2000	1.0151×10^4	872.9	11.63
Hysteresis control (time-dependent)	1000	315.43	1.5	210.29
Hysteresis control (time-independent)	1000	315.43	1.3	242.64
Hysteresis control, steady state (time-independent)	2000	1327	1.3	1.0208×10^3

[8] Z. Zhang *et al.*, "Efficient uncertainty quantification for the periodic steady state of forced and autonomous circuits," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 60, no. 10, pp. 687–691, Oct 2013.

[9] F. Lu *et al.*, "Limitations of polynomial chaos expansions in the bayesian solution of inverse problems," *J. Computational Physics*, vol. 282, pp. 138–147, Feb 2015.

[10] K. Konakli and B. Sudret, "Polynomial meta-models with canonical low-rank approximations: Numerical insights and comparison to sparse polynomial chaos expansions," *J. Computational Physics*, vol. 321, pp. 1144 – 1169, Jun 2016.

[11] M. Gerritsma *et al.*, "Time-dependent generalized polynomial chaos," *J. Computational Physics*, vol. 229, no. 22, pp. 8333–8363, Nov 2010.

[12] W. Huang *et al.*, "System accuracy analysis of the multiphase voltage regulator module," *IEEE Trans. Power Electron.*, vol. 22, no. 3, pp. 1019–1026, May 2007.

[13] M. del Casale *et al.*, "Selection of optimal closed-loop controllers for dc-dc voltage regulators based on nominal and tolerance design," *IEEE Trans. Ind. Electron.*, vol. 51, no. 4, pp. 840–849, Aug 2004.

[14] X. Luo, P. V. Shevchenko, and J. B. Donnelly, "Addressing the impact of data truncation and parameter uncertainty on operational risk estimates," *J. Operational Risk*, vol. 2, no. 4, pp. 3–27, Mar 2007.

[15] M. Althoff *et al.*, "Formal verification of phase-locked loops using reachability analysis and continuization," *ACM Commun.*, vol. 56, no. 10, pp. 97–104, Oct 2013.

[16] M. Althoff, "Formal and compositional analysis of power systems using reachable sets," *IEEE Trans. Power Syst.*, vol. 29, no. 5, pp. 2270–2280, Sep 2014.

[17] Toyota. (2014, Feb. 12) Defect information report (nhtsa recall 14v-053). [Online]. Available: http://www-odi.nhtsa.dot.gov/acms/cs/jaxrs/download/doc/UCM450071/RCDNN-14V053-0945.pdf

[18] ——. (2015, Jul. 15) Defect information report (nhtsa recall 15v-449). [Online]. Available: http://www-odi.nhtsa.dot.gov/acms/cs/jaxrs/download/doc/UCM482439/RCORRD-15V449-4622.pdf

[19] L. Ljung, *System Identification: Theory for the User*, 2nd ed. New Jersey, USA: Prentice-Hall, Inc., 1999.

[20] H. Abbas *et al.*, "Formal property verification in a conformance testing framework," in *Proc. ACM-IEEE 12th Int. Conf. Formal Methods and Models for Syst. Design*, Lausanne, 2014, pp. 155–164.

[21] H. Behjati *et al.*, "Alternative time-invariant multi-frequency modeling of pwm dc-dc converters," *IEEE Trans. Circuits Syst. I: Regular Papers*, vol. 60, no. 11, pp. 3069–3079, Nov 2013.

[22] J. Kimball and P. Krein, "Singular perturbation theory for dc-dc converters and application to pfc converters," *IEEE Trans. Power Electron.*, vol. 23, no. 6, pp. 2970–2981, Nov 2008.

[23] T. Henzinger *et al.*, "HyTech: A model checker for hybrid systems," in *Computer Aided Verification*, O. Grumberg, Ed. Berlin Heidelberg: Springer, Mar. 1997, pp. 460–463.

[24] G. Frehse, "PHAVer: Algorithmic verification of hybrid systems past HyTech," vol. 10, no. 3, Jun. 2008, pp. 263–279.

[25] J. Bengtsson *et al.*, "UPPAAL a tool suite for automatic verification of

real-time systems," in *Hybrid Systems III*. Berlin Heidelberg: Springer, Jun. 2005, pp. 232–243.

[26] S. Ratschan and Z. She, "Safety verification of hybrid systems by constraint propagation based abstraction refinement," in *Hybrid Systems: Computation and Control*, M. Morari and L. Thiele, Eds. Berlin Heidelberg: Springer, Mar. 2005, pp. 573–589.

[27] E. Asarin *et al.*, "The d/dt tool for verification of hybrid systems," in *Computer Aided Verification*, E. Brinksma and K. G. Larsen, Eds. Berlin Heidelberg: Springer, Sep. 2002, pp. 365–370.

[28] X. Chen *et al.*, "Flow*: An analyzer for non-linear hybrid systems," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, N. Sharygina and H. Veith, Eds. Springer Berlin Heidelberg, 2013, vol. 8044, pp. 258–263.

[29] G. Frehse *et al.*, "SpaceEx: Scalable verification of hybrid systems," in *Proc. 23rd Int. Conf. on Comput. Aided Verification*, Snowbird, UT, 2011, pp. 379–395.

[30] M. Miranda and A. Lima, "Formal verification and controller redesign of power electronic converters," in *Proc. IEEE Int. Symp. Indust. Electron.*, Ajaccio, France, 2004, pp. 907–912.

[31] T. Johnson *et al.*, "Design verification methods for switching power converters," in *Proc. 3rd Power and Energy Conf. at Illinois*, Urbana, IL, 2012, pp. 1–6.

[32] S. Hossain *et al.*, "Reachability analysis of closed-loop switching power converters," in *Proc. 4th Power and Energy Conf. at Illinois*, Urbana, IL, 2013, pp. 130–134.

[33] M. Hongbo and F. Quanyuan, "Hybrid modeling and control for buck-boost switching converters," in *Proc. Int. Conf. Commun., Circuits and Syst.*, Milpitas, CA, 2009, pp. 678–682.

[34] M. Senesky *et al.*, "Hybrid modelling and control of power electronics," in *Proc. 6th Int. Workshop on Hybrid Systems: Computation and Control*, Prague, Czech Republic, 2003, pp. 450–465.

[35] C. Sreekumar and V. Agarwal, "A hybrid control algorithm for voltage regulation in dcdc boost converter," *IEEE Trans. Ind. Electron.*, vol. 55, no. 6, pp. 2530 – 2538, Jun. 2008.

[36] Y. Quan *et al.*, "Simultaneous ccm and dcm operations of boost converter by a pwm hybrid control strategy," in *Proc. IEEE 39th Annual Conf. Ind. Electron. Society*, Vienna, Austria, 2013, pp. 1260–1265.

[37] U. Kuhne, "Analysis of a boost converter circuit using linear hybrid automata," ENS Cachan, Cedex, France, Tech. Rep., 2010.

[38] E. Hope *et al.*, "A reachability-based method for large-signal behavior verification of dc-dc converters," *IEEE Trans. Circuits Syst. I*, vol. 58, no. 12, pp. 2944–2955, Dec. 2011.

[39] T. Henzinger, "The theory of hybrid automata," in *Proc. IEEE Symp. on Logic in Comput. Science*, New Brunswick, NJ, 1996, pp. 278–292.

[40] N. Femia and G. Spagnuolo, "Genetic optimization of interval arithmetic-based worst case circuit tolerance analysis," *IEEE Trans. Circuits Syst. I: Fundam. Theory Appl.*, vol. 46, no. 12, pp. 1441–1456, Dec 1999.

[41] T. Ding *et al.*, "How affine arithmetic helps beat uncertainties in electrical systems," *IEEE Circuits Syst. Mag.*, vol. 15, no. 4, pp. 70–79, Fourthquarter 2015.

[42] R. Moore *et al.*, *Introduction To Interval Analysis*. Philadelphia, PA, USA: Soc. Ind. Appl. Math., 2009.

[43] J. Rohn, "Stability of interval matrices: the real eigenvalue case," *IEEE Trans. Autom. Control*, vol. 37, no. 10, pp. 1604–1605, Oct. 1992.

[44] M. Althoff *et al.*, "Analyzing reachability of linear dynamic systems with parametric uncertainties," in *Modeling, Design, and Simulation of Systems with Uncertainties*, A. Rauh and E. Auer, Eds. Berlin Heidelberg: Springer, May 2011, pp. 69–94.

[45] A. Donz and G. Frehse, "Modular, hierarchical models of control systems in spaceex," in *European Control Conf.*, 2013, pp. 4244–4251.

[46] A. Aziz *et al.*, "Efficient control state-space search," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 20, no. 2, pp. 332–336, Feb 2001.

[47] P. Hnsch *et al.*, "Reachability analysis of linear systems with stepwise constant inputs," *Electronic Notes in Theoretical Computer Science*, vol. 297, no. 0, pp. 61–74, Dec. 2013.

[48] *PLECS Manual Version 3.7*, Plexim Inc., Cambridge, MA, USA, 2015.

[49] *MATLAB Stateflow User's Guide*, Mathworks, MA, USA, 2015.

[50] S. Bak *et al.*, "HyST: A source transformation and translation tool for hybrid automaton models," in *Proc. ACM 18th Int. Conf. on Hybrid Syst.: Computation and Control*, Seattle, WA, 2015, pp. 128–133.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2017.2656905, IEEE Transactions on Industrial Informatics

Detection of False-data Injection Attacks in Cyber-Physical DC Microgrids

Omar Ali Beg, *Student Member, IEEE,* Taylor T. Johnson, *Member, IEEE,* and Ali Davoudi, *Senior Member, IEEE*

***Abstract*—Power electronics-intensive DC microgrids use increasingly complex software-based controllers and communication networks. They are evolving into cyber-physical systems (CPS) with sophisticated interactions between physical and computational processes, making them vulnerable to cyber attacks. This work presents a framework to detect possible false-data injection attacks (FDIA) in cyber-physical DC microgrids. The detection problem is formalized as identifying a change in sets of inferred candidate invariants. Invariants are microgrids properties that do not change over time. Both the physical plant and the software controller of CPS can be described as Simulink/Stateflow (SLSF) diagrams. The dynamic analysis infers the candidate invariants over the input/output variables of SLSF components. The reachability analysis generates the sets of reachable states (reach sets) for the CPS modeled as hybrid automata. The candidate invariants that contain the reach sets are called the actual invariants. The candidate invariants are then compared with the actual invariants, and any mismatch indicates the presence of FDIA. To evaluate the proposed methodology, the hybrid automaton of a DC microgrid, with a distributed cooperative control scheme, is presented. The reachability analysis is performed to obtain the reach sets and, hence, the actual invariants. Moreover, a prototype tool, HYbrid iNvariant GEneratoR (Hynger), is extended to instrument SLSF models, obtain candidate invariants, and identify FDIA.**

***Index Terms*—Cyber-physical systems, dc microgrid, distributed control, false-data injection attack, hybrid automaton.**

I. Introduction

ISLANDED multi-converter DC microgrids have advantages over their AC counterparts, including higher reliability, simpler control, and more efficient interfacing with naturally-DC renewable energy sources, electronics loads, and energy storage units [1], [2]. Therefore, DC microgrids have emerged as a key technology for the future, and their related control methodologies are also evolving. Given the well-established advantages of distributed control schemes over centralized control methodologies, the migration from current central controllers to future distributed schemes is inevitable [3]–[8]. The centralized control systems require two-way, high bandwidth communication links between the central controller and every other agent, and expose a single point-of-failure. Moreover, sparsity of communication networks utilized in distributed control schemes reduces the infrastructure cost, and improves solution scalability compared to a fully-connected communication network.

These DC microgrids are evolving into cyber-physical systems (CPS) with sophisticated software-based control and communication networks. Such CPS are, however, vulnerable to cyber attacks, as there is no central entity to monitor activities of all DC-DC converters leading to a limited global situational awareness. This vulnerability is analogous to the situation in cyber-physical power systems that have faced various types of cyber attacks, e.g., false-data injection attack (FDIA) [9], denial of service [10], [11], jamming [12], and random attacks [13]. Some prevention strategies for jamming include frequency hopping, direct-sequence spread spectrum technique, channel surfing, and protocol hopping [14]. In this work, detection of FDIA in power electroncis-intensive DC microgrids is considered that involves spoofing a signal, either in sensors or the communication network, through an attack vector that aims to disrupt the steady-state operation [9]. The attack vector formulation is a sophisticated process, and requires expert knowledge of the entire system. The intruder should have either physical access to a specified number of meters, or a complete knowledge of the infrastructure and the communication network [9].

The preventive measures against FDIA include physical security, information security, and communication security. With regards to the physical security, a minimum number of strategically selected set of sensor measurements (called as basic measurements) that need to be protected to thwart FDIA has been proposed [15]. Moreover, phasor measurement units (PMUs) can be strategically placed to protect power grids against such attacks [16]. However, PMUs are also vulnerable due to their use of global positioning systems [17]. With regards to information security, a prevention strategy against FDIA involves dynamically changing the information structure of microgrids [18]. In general, the communication security can be improved using stringent cryptographic techniques, i.e., encryption, authentication, and key management for power systems [19]. For example, a communication security architecture for distributed microgrid control [20] exchanges encrypted information. A trusted sensing base is proposed in the form of a current transformer that encrypts the AC power signal before sending it to PMUs [21].

Recent work on FDIA detection, albeit in power systems [11], [13], [22]–[29], broadly employs state estimation processes, e.g., using Kalman filters [13], sparse op-

The material presented in this paper is based upon work supported by the National Science Foundation (NSF) under grant numbers ECCS-1405173, CNS 1464311, and SHF 1527398, the Air Force Research Laboratory through contract number FA8750-15-1-0105, and the Air Force Office of Scientific Research under contract numbers FA9550-15-1-0258 and FA9550-16-1-0246. Omar A. Beg, and A. Davoudi are with the University of Texas, Arlington, TX 76019, USA. T. Johnson is with Vanderbilt University, Nashville, TN 37240, USA. (e-mail: omar.beg@mavs.uta.edu; taylor.johnson@vanderbilt.edu; davoudi@uta.edu).

Manuscript received August 29, 2016; revised December 22, 2016; accepted January 12, 2017.

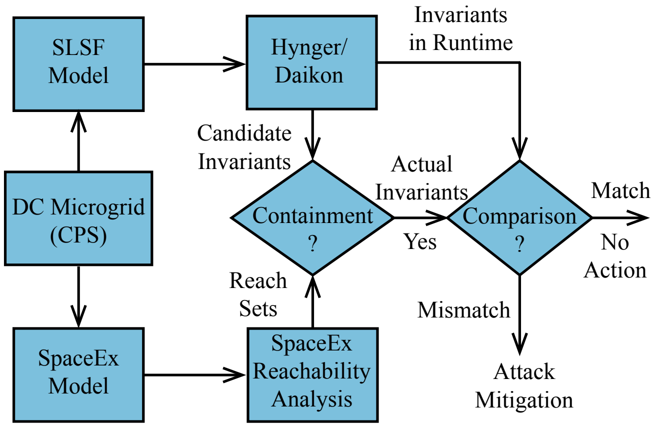


Fig. 1. The proposed FDIA detection framework bridges the gap between the software-based anomaly detection techniques and power electronics-intensive DC microgrids modeled as hybrid automata.

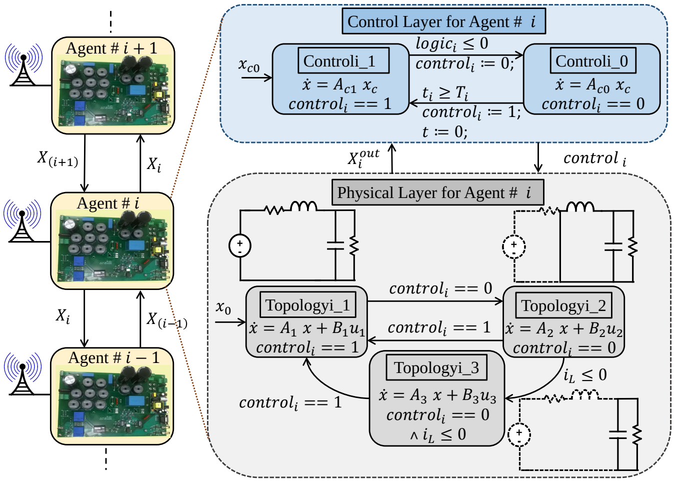


Fig. 2. Hybrid automaton of a cyber-physical DC microgrid showing converter and controller interactions. Each converter, its corresponding controller, and its communication links are, altogether, considered as an agent. Each agent shares its information with the neighboring agents on the communication graph.

timization [22], generalized likelihood ratio [23], Kullback-Leibler distance [24], Chi-square detector and similarity matching [25], state forecasting [26], and machine-learning techniques [27]. However, to the best of authors' knowledge, FDIA detection in software-intensive DC microgrids is not systematically studied yet. This work aims to formalize the FDIA detection problem as a change in sets of inferred *invariants*; system properties that do not change over time. Here, invariants are defined in terms of bounds over the output voltage and current of individual converters.

The overall block diagram of the proposed FDIA detection framework is shown in Fig. 1. The candidate invariants are inferred from the Simulink/Stateflow (SLSF) model of the DC microgrid. Hynger (HYbrid iNvariant GEneratoR) [30] tool is used to provide an interface between the SLSF model and the Daikon tool [31], [32]. Daikon is a software-based invariant inference tool. Hynger takes the SLSF model as an input, executes it to generate time traces, and transforms them into a format compatible with Daikon to generate candidate invariants. Moreover, the cyber-physical DC microgrid is formally modeled as multi-agent hybrid automata, and the reachability analysis is performed using SpaceEx [33] to obtain the reachable set of states (called the reach sets). The Hynger/Daikon combination provides only the candidate invariants. The SpaceEx tool is used concurrently in the proposed framework to obtain the actual invariants. The candidate invariants that contain the reach sets are called the actual invariants. The candidate invariants are then compared with the actual invariants, and any mismatch indicates the presence of FDIA. A mitigation strategy can then disconnect the affected converter and prevent the microgrid's instability.

The remainder of this paper is organized as follows: The hybrid automaton modeling of DC microgrids that includes both physical and cyber layers is discussed in Section II. The FDIA detection framework for DC microgrids is discussed in Section III. Section IV studies a DC microgrid prototype, with an analysis of FDIA effects, detection using the proposed framework, and mitigation. Section V concludes the paper.

II. Cyber-physical DC Microgrid as Multi-agent Hybrid Automata

The proposed FDIA detection framework requires the CPS modeled as SLSF diagrams and as hybrid automata to obtain the candidate invariants and the reach sets, respectively. A hybrid automaton [34] is a formal model, essentially a finite-state machine with additional continuous dynamic variables. Cyber-physical DC microgrids can be modeled as multi-agent hybrid automata, where power electronics DC-DC converters (referred to as converters) form the physical layer, and the software-based controller with the communication network among converters, altogether, form the cyber layer. Each converter, with its corresponding controller and communication links, is considered an agent, and its hybrid automaton is shown in Fig. 2. This hybrid automaton exchanges information with its two immediate neighbors, e.g., $(i+1)$th and $(i-1)$th agents in Fig. 2, through *global variables* to implement a cooperative control protocol.

A. Modeling the Physical Layer

The output voltage v_i^{out} and output current i_i^{out} of the ith converter are regulated by controlling the MOSFET switch through the corresponding control layer. The switching state of the MOSFET switch leads three different topologies (switching sub-interval) as shown in Fig. 2. The state of a hybrid automaton may change either through a continuous flow trajectory within a given topology, or through a discrete transition between two given topologies.

1) Formal Hybrid Automaton: Let $\mathbb{R}^n$ be the set of n-dimensional reals, and 2^X be the power set of a given set X, i.e., the set of all the subsets of X.

Definition 2.1: A *hybrid automaton* is defined by a tuple $\mathcal{H} = \langle Q, X, \Theta, U, F, \mathcal{T}, E, G, inv \rangle$:

- $Q = \{q_1, q_2, ..., q_N\}$ is a finite set of topologies.
- X is a finite set of continuous variables, with $\forall\ x \in X\ \exists\ val(x) \in \mathbb{R}$, where $val(x)$ is a valuation of x as a result of a function mapping. $X = X_g \cup X_l$, such that X_g is the set of global variables and X_l is the set of local

variables. Further, $X_g = In \cup O$, where In is the set of global input variables and O is the set of global output variables. A *state* is defined by $s = (q, val(x)) \in Q \times \mathbb{R}$.
- $\Theta \subseteq Q \times \mathbb{R}^n$ is a set of initial conditions.
- $U = \{u_1, u_2, ...u_N\}$ is the set of inputs for each topology.
- F is a finite set of ODEs defined for each $q \in Q$ over the continuous variables $x \in X$. $F(q, x)$ defines the continuous dynamics for each $q \in Q$ over a time period T, and assigns a Lipschitz continuous vector space in $\mathbb{R}^n$.
- $\mathcal{T}$ is a finite set of continuous flow trajectories that define $val(x)$ over $[0\ T]$ from given initial conditions $(q, x_0) \in \Theta$, such that $\forall\ \tau(q, x) \in \mathcal{T}$, $\exists\ s \in \tau(q, x)$ that satisfies $inv(q)$ (i.e., $s \in \tau(q, x) \vDash inv(q)$). A continuous flow trajectory is given by
$$\tau(q, x) = x_0 + \int_0^T F(q, x)dt. \qquad (1)$$
- E is a finite set of feasible discrete transitions allowed among the topologies. It is defined by a tuple $e = \langle q, q', g, x' \rangle$, such that a discrete transition is allowed from source topology q to the destination topology q' only when the associated guard condition g is satisfied, and the continuous state is updated to x' after the transition. It might not be possible to visit the entire set of topologies from one particular topology.
- $G \subseteq 2^X$ is the guard set such that $\forall\ e\ \exists\ g \in G$. A *guard* must be satisfied by a state to take a discrete transition from a given topology to another. A state $s = (q_k, val(x))$ *satisfies* g (i.e, $s \vDash g$) iff $q_k = q_l \in e = (q_l, q_l', g, x')$ and $val(x) \in g$.
- inv is a finite set of invariants, where an invariant is associated to each given topology, i.e., $\forall\ q \in Q\ \exists\ inv(q) \subseteq \mathbb{R}^n$. An *invariant* is a property of the hybrid automaton that must be satisfied by all the states for a given topology. A state $s \vDash inv(q)$ iff $val(x) \in inv(q)$.

If a state $(q, val(x))$ does not satisfy an invariant $inv(q)$, the continuous state x stops evolving within a topology. The guard function ensures a discrete transition to an appropriate topology once the corresponding guard is satisfied. Here, invariants and guards are defined in the form of bounds over continuous state variables. The semantics of the hybrid automaton $\mathcal{H}$ is defined by its execution, ϵ. An *execution* is defined as a sequence of states, $\epsilon = s_0, s_1, s_2, \ldots$, obtained as a result of continuous flow trajectories and discrete transitions.

2) Instantiation of the Physical Layer: The hybrid automaton of the ith buck converter is considered for instantiation, where v_i^{in} is the DC input voltage. The continuous dynamics, for a given topology, is given by a set of state-space equations
$$\frac{dx}{dt} = A_q x + B_q u. \qquad (2)$$
$A_q \in \mathbb{R}^{n \times n}$ and $B_q \in \mathbb{R}^{n \times m}$ are system matrices. Subscript q denotes the appropriate topology. The instantiation of the hybrid automaton for the ith agent, as per Definition 2.1, is
- Three topologies, shown in Fig. 2, are denoted by $Q = \{q_1, q_2, q_3\}$.
- $X = \{i_i^L, v_i^C, i_i^{out}, v_i^{out}, control_i\}$, where $X_l = \{i_i^L, v_i^C\}$ and $X_g = \{i_i^{out}, v_i^{out}, control_i\}$.
- $U = \{[v_i^{in}, 0, 0, 0]', [0, 0, 0, 0]', [0, 0, 0, 0]'\}$ forms the input vector set.
- $E = \{(q_1, q_2, g_{12}, x'), (q_2, q_1, g_{21}, x'), (q_2, q_3, g_{23}, x'), (q_3, q_1, g_{31}, x')\}$ defines the feasible discrete transitions, e.g., (q_2, q_3, g_{23}, x') means that a discrete transition from topology q_2 to q_3 is allowed, if the guard $g_{23} = \{(i_i^L \leq 0)\}$ is satisfied and the continuous state is reset to x'.
- Guard set, for the corresponding elements of E, is defined by $G = \{(control_i == 0), (control_i == 1), (i_i^L \leq 0), (control_i == 1)\}$. Signals received from the control layer are $control_i == 1$ and $control_i == 0$ to set the MOSFET ON and OFF, respectively.
- The continuous flow trajectory is defined by (2), with the corresponding state matrices for each topology.

The evolution of the hybrid automaton model starts with initial conditions from the set $init$, e.g., $(q_1, x_0) \in init$ for a given input $u_1 = [v_i^{in}, 0, 0, 0]'$ and, subsequently, the continuous state evolves according to the flow function. The topology remains the same, i.e., $q(t) = q_1$, as x_0 evolves inside the invariant $inv(q_1)$ and attains a final value $x' \in inv(q_1)$. Once the continuous state x' satisfies the corresponding guard, $g_{12} = \{(control_i == 0)\}$ corresponding to the topology q_1, the topology may transition from q_1 to q_2, and the continuous state is reset with a new value x'' in the new invariant set $inv(q_2)$ with a new input $u_2 = [0, 0, 0, 0]'$.

B. Modeling the Cyber Layer

Microgrid control hierarchy is divided into three levels, i.e., primary, secondary, and tertiary [35]. Primary control features the fastest response, and is based entirely on local measurements with no communication. Secondary control operates on a slower time scale, often with reduced communication bandwidth by using sampled measurements. In this work, we consdier two control objectives: proportional load sharing among converters, according to their power ratings, and global voltage regulation of the distribution bus. These objectives are implemented in the secondary control layer through proportional load sharing sub-layer and global voltage regulation sub-layer (which includes a voltage observer and a noise cancellation module), as shown in Fig. 3. We use a distributed cooperative control scheme, i.e., the output of a particular agent depends only on its information and its N_i neighbors on the communication graph [3]. A *graph* $\mathbb{G}$ is defined as a pair (tuple) of a set of vertices and edges, i.e., $\mathbb{G} = (\Lambda, \varepsilon)$. Let $\Lambda = \{\lambda_1, \lambda_2, ..., \lambda_N\}$ define a set of N vertices (nodes), and $\varepsilon \subseteq \Lambda \times \Lambda$ a set of edges. An edge from node λ_i to λ_j is a pair $(\lambda_i, \lambda_j) \in \varepsilon$. The graph is said to be *bi-directional* if $(\lambda_i, \lambda_j) \in \varepsilon \implies (\lambda_j, \lambda_i) \in \varepsilon,\ \forall\ i, j \in \Lambda$.

A graph may be represented by an *adjacency matrix* $\mathbb{A} = [a_{ij}]$ with weights $a_{ij} > 0$ if $(\lambda_j, \lambda_i) \in \varepsilon$, and $a_{ij} = 0$ otherwise. The local control protocol, u_i for each agent i is
$$u_i = \sum_{j \in N_i} a_{ij}(x_j - x_i), \qquad (3)$$
such that the control of each agent depends only on the difference between its state and those of its neighbors. This protocol ensures that all agents reach a consensus.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2017.2656905, IEEE Transactions on Industrial Informatics

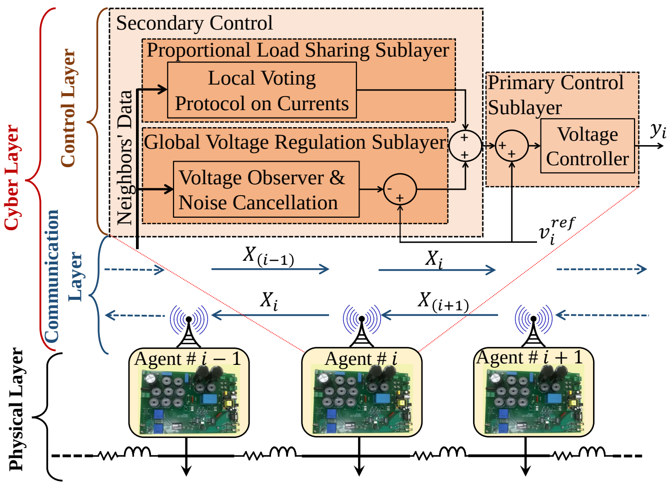


Fig. 3. Structure of the cyber-physical DC microgrid showing the cyber and physical layers and the control sub-layers.

The global voltage reference for N agents is defined as $V_{ref} = [v_1^{ref}, v_2^{ref}, ..., v_N^{ref}]^T$, input DC voltage as $V_{in} = [v_1^{in}, v_2^{in}, ..., v_N^{in}]^T$, output DC voltage vector as $V_{out} = [v_1^{out}, v_2^{out}, ..., v_N^{out}]^T$, output current vector as $I_{out} = [i_1^{out}, i_2^{out}, ..., i_N^{out}]^T$, the voltage estimation vector for the voltage observer module as $V_{est} = [v_1^{est}, v_2^{est}, ..., v_N^{est}]^T$, the per-unit current vector as $X_{pu} = [x_1^{pu}, x_2^{pu}, ..., x_N^{pu}]^T$, and the estimate of the voltage deviation vector for the noise cancellation module as $W_{est} = [w_1^{est}, w_2^{est}, ..., w_N^{est}]^T$. Here, x_i^{pu} refers to the loading percentage of the ith agent. As shown in Fig. 3, X_i depicts the information vector communicated from the ith agent to the $(i-1)$th and $(i+1)$th agents, such that $X_i = [x_i^{pu}, v_i^{est}, w_i^{est}]^T$. Moreover, X_{i-1}, and X_{i+1} of Fig. 3 are defined similarly. Communication links are modeled as low-pass filters ($\mathbb{T}_1$ and $\mathbb{T}_2$ in Appendix) to emulate delays inherent in the data exchange process, as in [3], [36], [37]. Here, i_i^{out} and v_i^{out} are passed through $\mathbb{T}_1$ to get the per-unit current x_i^{pu} and the voltage y_i^{vo}, respectively.

The control sub-layers are discussed next.

1) Proportional load sharing sub-layer: The *ith* agent shares per-unit current information with its immediate neighbors, i.e., $(i-1)$th and $(i+1)$th agents. This sub-layer has a PI controller with parameters $P(i,i,)$ and $I(i,i)$, where P and I are $N \times N$ matrices that contain the proportional and integral terms, respectively. If $\mathbb{C}$ is the adjacency matrix for the cooperative control strategy, the per-unit current information from $(i-1)$th and $(i+1)$th agents communicated to the PI controller is processed as

$$x_{i-1\to i}^{pu} = \left(x_{i-1}^{pu} - x_i^{pu}\right).\mathbb{C}(i,i-1) \tag{4}$$

and

$$x_{i+1\to i}^{pu} = \left(x_{i+1}^{pu} - x_i^{pu}\right).\mathbb{C}(i,i+1), \tag{5}$$

respectively. Let the state variable of the PI controller be x_i^i, then the corresponding ODE is given by

$$\dot{x}_i^i = \left(x_{i-1\to i}^{pu} + x_{i+1\to i}^{pu}\right).I(i,i). \tag{6}$$

The output, v_i^i, of this layer is given by

$$v_i^i = \left(x_{i-1\to i}^{pu} + x_{i+1\to i}^{pu}\right).P(i,i) + x_i^i, \tag{7}$$

which is passed to the primary control sub-layer.

2) Global voltage regulation sub-layer: If $\mathbb{A}$ is the adjacency matrix for the cooperative control strategy, the voltage estimation information from $(i-1)$th and $(i+1)$th agents is further processed as

$$v_{i-1\to i}^{est} = \left(v_{i-1}^{est} - v_i^{est}\right).\mathbb{A}(i,i-1) \tag{8}$$

and

$$v_{i+1\to i}^{est} = \left(v_{i+1}^{est} - v_i^{est}\right).\mathbb{A}(i,i+1) \tag{9}$$

respectively. This voltage estimate is then passed through an integrator, with the state variable v_i^{esti}, such that

$$\dot{v}_i^{esti} = \left(v_{i-1\to i}^{est} + v_{i+1\to i}^{est}\right). \tag{10}$$

In the noise-cancellation module, the ith agent shares the estimate information of the voltage deviation w_i^{est} with its immediate neighbors. The actual voltage deviation for the ith agent is

$$w_i = \left(v_i^{est} - v_i^{out}\right). \tag{11}$$

If $\mathbb{B}$ is the adjacency matrix for the cooperative control strategy, the information about the estimate of the voltage deviation from $(i-1)$th and $(i+1)$th agents is

$$w_{i-1\to i}^{est} = \left(w_{i-1}^{est} - w_i^{est}\right).\mathbb{B}(i,i-1) \tag{12}$$

and

$$w_{i+1\to i}^{est} = \left(w_{i+1}^{est} - w_i^{est}\right).\mathbb{B}(i,i+1), \tag{13}$$

respectively. This estimate is passed through an integrator, with the state variable w_i^{esti}, such that

$$\dot{w}_i^{esti} = \left(w_{i-1\to i}^{est} + w_{i+1\to i}^{est}\right). \tag{14}$$

The estimate for the voltage deviation, w_i^{est}, is

$$w_i^{est} = w_i + w_i^{esti}. \tag{15}$$

This estimate is then passed to a second integrator with a gain K of dimension $N \times N$, and with the state variable w_i^{estii},

$$\dot{w}_i^{estii} = w_i^{est}. \tag{16}$$

The average voltage of the microgrid as estimated by the ith agent, based on the neighbor information, is

$$v_i^{avg} = v_i^{est} = v_i^{esti} + v_i^{out} - w_i^{estii}.K(i,i). \tag{17}$$

This sub-layer has a PI controller with parameters $P(i,i,)$ and $I(i,i)$. The difference between the global reference voltage and the global average voltage as determined by the ith agent is passed through this PI controller. Let the state variable for PI controller be denoted by v_i^{avgi}, then the ODE is given by

$$\dot{v}_i^{avgi} = (v_i^{ref} - v_i^{avg}).I(i,i). \tag{18}$$

The voltage regulation term at the controller output is

$$v_i^{greg} = v_i^{avgi} + (v_i^{ref} - v_i^{avg}).P(i,i). \tag{19}$$

3) Primary control sub-layer: There is a PI controller with parameters P_{mc} and I_{mc}, and a transfer function $\mathbb{T}_2$. The output of $\mathbb{T}_2$ is denoted by y_i^{mc}. The input u_i^{mc} is

$$u_i^{mc} = v_i^{ref} + v_i^i + v_i^{greg}. \tag{20}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2017.2656905, IEEE Transactions on Industrial Informatics

The expression for v_i^i and v_i^{greg} are given by (7) and (19), respectively. The ODE for the state variable x_i^{pi} associated with the PI controller is given by

$$\dot{x}_i^{pi} = \left(y_i^{mc} - y_i^{vo}\right).I_{mc}. \tag{21}$$

The output of this sub-layer, y_i, is given by

$$y_i = P_{mc}.(y_i^{mc} - y_i^{vo}) + x_i^{pi}, \tag{22}$$

that drives the MOSFET of the ith converter. The cyber layer has two topologies, i.e., `controli_1` and `controli_0`, as shown in Fig. 2, to generate control signal, $control_i$. It may evaluate to $controli == 1$ and $controli == 0$, that correspond to 'ON' and 'OFF' pulses for the MOSFET, respectively. The hybrid automaton generates $controli == 1$ in `controli_1`, and $controli == 0$ in `controli_0`. The ODEs developed for the cyber layer of DC microgrid above are used to describe the continuous dynamics for the two topologies. The switching logic, $logic_i$, is formulated using (22), the elapsed time t_i, and the time period T_i of the ith agent. This is implemented in the hybrid automaton model as a guard to enforce the discrete transition from topology `controli_1` to the topology `controli_0`, hence generating control signal $controli == 0$. Whereas, transition from `controli_0` to `controli_1` is entirely dependent upon the time period T_i that forms the corresponding guard to ensure a periodic switching. This transition is enforced by the guard $t_i \geq T_i$, hence generating control signal $controli == 1$.

4) Instantiation of the cyber layer: The instantiation of the hybrid automation model for the cyber layer of the ith agent, as per Definition 2.1, is

- Two topologies are denoted by $Q = \{q_4, q_5\}$.
- The continuous state vector is $X = X_l \cup X_g$, where, $X_l = \{x_i^i, v_i^{esti}, w_i^{esti}, v_i^{avgi}, x_i^{pi}\}$ and $X_g = \{x_i^{pu}, v_i^{est}, w_i^{est}, i_i^{out}, v_i^{out}, x_{i+1}^{pu}, v_{i+1}^{est}, w_{i+1}^{est}, x_{i-1}^{pu}, v_{i-1}^{est}, w_{i-1}^{est}, control_i\}$.
- $E = \{(q_4, q_5, g_{45}, x'), (q_5, q_4, g_{54}, x')\}$ defines the feasible discrete transitions, e.g., (q_5, q_4, g_{54}, x') means a discrete transition from the topology q_5 to q_4 is allowed, if the guard $g_{54} = \{(t_i \geq T_i)\}$ is satisfied and the continuous state is reset to a new value x'.
- Guard set, for the corresponding elements of E, is defined by $G = \{(logic_i \leq 0), (t_i \geq T_i)\}$.
- The continuous flow trajectory is given by ODEs in (6), (10), (14), (16), (18), and (21) for both topologies.

The control layer and the physical layer both interact with each other and exchange $control_i$ and X_i^{out} as shown in Fig. 2, where $X_i^{out} = [v_i^{out}, i_i^{out}]^T$ and $control_i$ drives the switching in the physical layer. A 50 μs fixed time-step for the numerical solver in the Simulink, and 4 μs sampling time are used in the dSPACE platform.

C. Hybrid Input/Output Automata Conditions

The closed-loop control systems are modeled using hybrid input/output automata (HIOA), to form as a singleton hybrid automaton [38]. Here, the converter and the controller are modeled as two hybrid automata, interacting with each other in a parallel composition, provided that their local variables are disjoint from each other and the two automata are compatible.

Definition 2.2: Let $\mathcal{H}_{ip}$ and $\mathcal{H}_{ic}$ denote the hybrid automata of the converter and the controller for the ith agent, respectively. They are *compatible* if they meet following three conditions

1) $In_{ip} \subseteq O_{ic} \cup O_{(i+1)p} \cup O_{(i-1)p}$,
2) $In_{ic} \subseteq O_{ip} \cup O_{(i+1)c} \cup O_{(i-1)c}$, and
3) $O_{ip} \cap O_{ic} \cap O_{(i+1)c} \cap O_{(i+1)p} \cap O_{(i-1)c} \cap O_{(i-1)p} = \varnothing$.

Subscripts p and c denote the plant (i.e., converter) and the controller, respectively.

The corresponding input and output variables, for the ith agent, are

$$\begin{cases} In_{ip} = \{control_i\}, \\ In_{ic} = \{v_i^{out}, i_i^{out}, X_{i-1}, X_{i+1}\}, \\ O_{ip} = \{v_i^{out}, i_i^{out}\}, \\ O_{ic} = \{control_i, X_i\}. \end{cases} \tag{23}$$

The output variables for the $(i+1)$th and $(i-1)$th agents are

$$\begin{cases} O_{(i+1)p} = \{v_{i+1}^{out}, i_{i+1}^{out}\}, \\ O_{(i+1)c} = \{control_{i+1}, X_{i+1}\}, \\ O_{(i-1)p} = \{v_{i-1}^{out}, i_{i-1}^{out}\}, \\ O_{(i-1)c} = \{control_{i-1}, X_{i-1}\}. \end{cases} \tag{24}$$

It is obvious that the ith agent (comprising converter and controller) meets the compatibility conditions of Definition 2.2, and a parallel composition can be formed. For the ith agent, the parallel composition is $\mathcal{H}_i = \mathcal{H}_{ip} \parallel \mathcal{H}_{ic}$. The DC microgrid is a parallel composition of N agents, i.e, $\mathcal{H}_1 \parallel \mathcal{H}_2 \parallel \cdots \parallel \mathcal{H}_i \parallel \cdots \parallel \mathcal{H}_N \parallel \mathcal{H}_1$, where $\forall\ i\ \mathcal{H}_i = \mathcal{H}_{ip} \parallel \mathcal{H}_{ic}$.

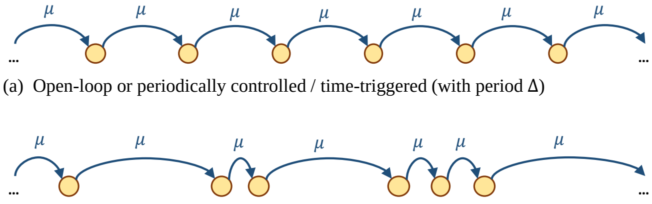
III. FDIA Detection Using Hynger

A. FDIA Scenario Formulation

In cyber-physical DC microgrids, the information among the agents is shared through the global variables (e.g., X_i) that are vulnerable to the FDIA. An FDIA mixes the original data/measurements vector with a malicious vector. The intruder may target the global variables and the sensors data to disturb the consensus procedure, as will be demonstrated in Section IV. In an unconstrained scenario, the intruder has access to all the global variables, and may randomly select some (or all). Under constrained FDIA, the intruder has limited access to one or a few global variables, and formulates the FDIA vector to target these. Let $X_g \in \mathbb{R}^k$ be the vector containing the global variables. FDIA vector $W \in \mathbb{R}^k$ may be formulated to obtain the compromised vector $Z = X_g + \alpha W$, where α is a real valued multiplicative factor that defines the *weight* of the FDIA vector. Each element of the FDIA vector is denoted by w_i, such that a nonzero entry signifies that the corresponding global variable in X_g is targeted. For unconstrained FDIA, all elements of $W \in \mathbb{R}^k$ are nonzero.

B. Hynger - An Overview

Hynger is a MATLAB-based software tool to produce invariants for cyber-physical systems modeled using SLSF. Hynger uses MATLAB's application program interfaces (APIs) to interact with SLSF models during simulations [30], and inserts instrumentation points for selected state variables. Instrumentation points may be regarded as the observation



(a) Open-loop or periodically controlled / time-triggered (with period Δ)

(b) Aperiodically controlled / event-triggered

Fig. 4. The instrumentation points are added by Hynger into open-loop, periodically or aperiodically controlled SLSF models.

points to record state variable values at each simulation time-step. Hynger can instrument both open-loop or periodic, and aperiodic control actions, μ, as shown in Fig. 4. The instrumentation points are inserted into the SLSF diagram using function calls through following two types of callbacks:

1) Precondition Callback: It is called before a Simulink block output method executes, i.e., the valuation of the state variable is recorded before the Simulink block execution. Hence, the state valuation is recorded at time t.

2) Postcondition Callback: It is called after a Simulink block output method executes, i.e., the valuation of a state variable is recorded after the Simulink block execution. Hence, the valuation of the state is recorded at time $t + \delta$, where δ is the simulation time-step.

As the SLSF diagram is simulated using Hynger, these instrumentation callback functions automatically insert the instrumentation points to generate a trace file format compatible with Daikon, a dynamic analysis tool used to generate likely invariants for software programs [31]. The analysis performed on a software program by actually executing it on a host processor is called the *dynamic analysis*. As the computational overhead for Hynger grows linearly with the number of monitored state variables [30], the user can select fewer state variables for monitoring (e.g., the output voltages and currents in DC microgrid) to reduce the computational overhead. Moreover, instead of selecting the entirety of the Simulink model for instrumentation, the user can select fewer Simulink blocks to further reduce the performance overhead.

C. FDIA Detection Framework

This framework involves inferring and checking sets of invariants to determine if an FDIA is underway. While this builds on the Hynger tool, extensions will be required to execute the tool and analyze results at runtime. The FDIA detection framework is shown in Fig. 5. A CPS model is provided as an SLSF diagram $\mathcal{A}$. The SLSF diagram is instrumented (denoted as $\hat{\mathcal{A}}$) using the Hynger tool, and is executed to generate a set of sampled, finite-precision traces $\mathscr{T}$ for given initial condition $\theta \in \Theta$. This adds instrumentation points for every input and output signal in the SLSF diagram. These generated traces are in Daikon compatible format that are passed on to Daikon, and analyzed to generate a set of candidate invariants $\hat{\Phi}$. However, Hynger/Daikon combination provides only the candidate invariants when used as standalone invariant generation tool. Each element $\hat{\varphi} \in \hat{\Phi}$ is then checked as actual invariant using the reachability analysis. The SpaceEx reachability analysis tool [33] is used to obtain the actual invariants. Changes in $\hat{\Phi}$ over time indicates an FDIA.

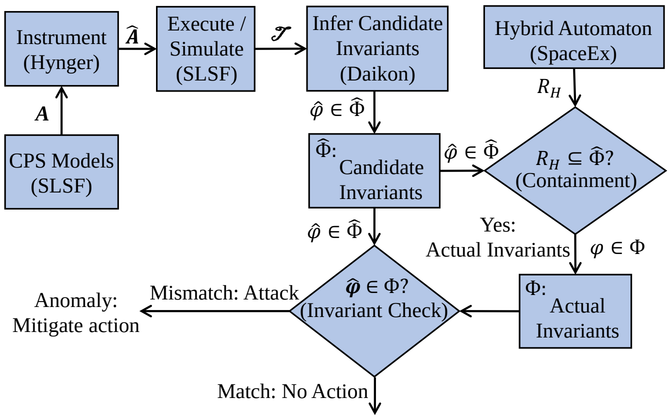

Instrument (Hynger)
$\hat{A}$
Execute / Simulate (SLSF)
$\mathscr{T}$
Infer Candidate Invariants (Daikon)
Hybrid Automaton (SpaceEx)
A
R_H
CPS Models (SLSF)
$\hat{\varphi} \in \hat{\Phi}$
$\hat{\Phi}$: Candidate Invariants
$\hat{\varphi} \in \hat{\Phi}$
$R_H \subseteq \hat{\Phi}$? (Containment)
$\hat{\varphi} \in \hat{\Phi}$
Yes: Actual Invariants
$\varphi \in \Phi$
Anomaly: Mitigate action
Mismatch: Attack
$\hat{\varphi} \in \Phi$? (Invariant Check)
Φ: Actual Invariants
Match: No Action

Fig. 5. FDIA detection framework using Hynger/Daikon to infer the candidate invariants and using SpaceEx reachability analysis to generate the reach sets.

For a given formal hybrid automaton $\mathcal{H}$ of a CPS, following definitions are introduced to extract the actual invariants from candidate invariants, as shown in Fig. 5:

Definition 3.1: For a hybrid automaton $\mathcal{H}$, all states encountered during executions are called the *reachable states* of $\mathcal{H}$. A state is already defined in Definition 2.1. Since the exact set of all reachable states is undecidable, reachability analysis tools compute the overapproximated sets of reachable states (called the *reach sets* for simplicity). In this work, SpaceEx [33] is used to compute the reach sets for a formal hybrid automaton $\mathcal{H}$, denoted by $\mathcal{R}_{\mathcal{H}}$.

Definition 3.2: The *property* ρ of a hybrid automaton $\mathcal{H}$ is defined as a Boolean-valued expression, that contains some or all state variables of $\mathcal{H}$, and evaluates to $True$ or $False$.

Definition 3.3: For a hybrid automaton $\mathcal{H}$, a state s is said to *satisfy* the property ρ (i.e., $s \models \rho$) if ρ evaluates to $True$ when all state variables are assigned values as defined by the state s.

Definition 3.4: For a hybrid automaton $\mathcal{H}$, a property ρ is an *invariant* of $\mathcal{H}$ if all its reach sets satisfy ρ, i.e., $\mathcal{R}_{\mathcal{H}} \models \rho$. A candidate invariant $\hat{\varphi} \in \hat{\Phi}$ is also a property of $\mathcal{H}$. Therefore, Definition 3.4 infers the actual invariants $\varphi \in \Phi$.

Definition 3.5: A candidate invariant $\hat{\varphi} \in \hat{\Phi}$ of a hybrid automaton $\mathcal{H}$ is the *actual invariant* $\phi \in \Phi$ iff $\mathcal{R}_{\mathcal{H}} \models \hat{\varphi} \in \hat{\Phi}$.

The candidate invariants for DC microgrids are obtained from Hynger in forms of bounds over the continuous state variables, and denoted as $[\mathcal{B}_l,\ \mathcal{B}_u]$. It is assumed that SLSF model depicts the hybrid automaton so that Hynger can find the set of candidate invariants $\hat{\Phi}$. Each $\hat{\varphi} \in \hat{\Phi}$ is then examined to ascertain whether it is an actual invariant as per Definition 3.5, i.e., checking whether $\mathcal{R}_{\mathcal{H}} \subseteq \hat{\varphi}$ holds.

The FDIA tends to disturb the consensus and hence the invariants as shown in case studies in Section IV. This change is employed to detect FDIA on DC microgrids.

Definition 3.6: A hybrid automaton $\mathcal{H}$ is said to be operating under FDIA scenario iff $\hat{\varphi} \notin \Phi$.

IV. Case Studies

A small-scale DC microgrid prototype is shown in Fig. 6, with the system parameters given in the Appendix. A comparison of the SLSF model simulation and the experimental data

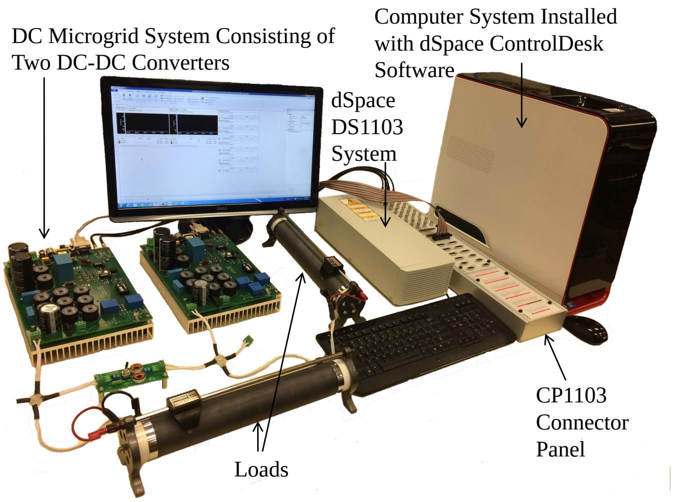

DC Microgrid System Consisting of Two DC-DC Converters
Computer System Installed with dSpace ControlDesk Software
dSpace DS1103 System
CP1103 Connector Panel
Loads

Fig. 6. Experimental setup for a DC microgrid consisting of two dc-dc converters and a dSpace DS1103 controller system.

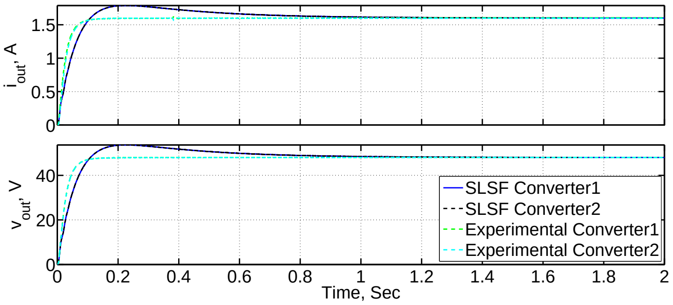


Fig. 7. SLSF simulation and experimental results for DC microgrid under no FDIA scenario showing stable output current and output voltage.

is shown in Fig. 7, for a stable output under no FDIA scenario. The effects of constrained FDIA on global variables, e.g., v_2^{est}, and w_2^{est} are shown in Fig. 8 and Fig. 9, respectively. The intruder may also disturb the consensus protocol when the current and voltage sensors are targeted as shown in Fig. 10 and Fig. 11, respectively. Unconstrained FDIA that involves targeting the entire set of global variables, is very effective in destabilizing the DC microgrid, as shown in Fig. 12.

A. FDIA Detection

For FDIA detection, the SLSF model formed using the methodology in Section II is instrumented using Hynger, and then simulated within the SLSF environment to generate traces under no FDIA scenario. This process generates the trace files, in Daikon compatible format, that are passed on to Daikon. Hence, the corresponding invariants are generated automatically, and shown in Table I. The SpaceEx reachability analysis tool computes the reach sets in the steady state, as seen in Fig. 13. It is shown that the experimental data and the simulation traces are contained within the reach sets. Moreover, reach sets satisfy the candidate invariants generated using Hynger under no FDIA scenario. Therefore, the invariants without FDIA of Table I are found to be the actual invariants as per Definition 3.5.

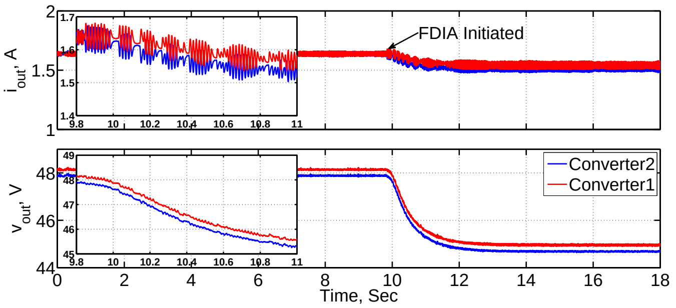


Fig. 9. Experimental data for the constrained FDIA, targeting w_2^{est}.

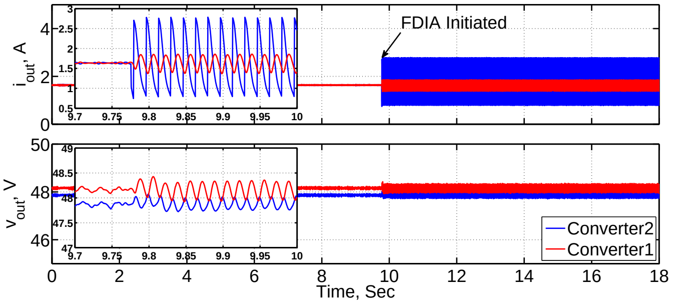


Fig. 10. Experimental data for the constrained FDIA targeting current sensor.

Next, FDIA detection approach is tested when the adversary breaks into the communication link from agent 2 to agent 1. A false data signal is spoofed into x_2^{pu}, at time $t = 0.6\ s$, through the compromised communication link. x_2^{pu} is the per-unit current information of agent 2 that is communicated to agent 1. The DC microgrid under FDIA scenario is again instrumented using Hynger, and simulated in the SLSF environment to generate traces and the corresponding invariants. The output of the instrumented model under FDIA is plotted in Fig. 14 for both agents 1 and 2. It can be observed that the consensus protocol is disturbed under FDIA.

The corresponding invariants for the DC microgrid under

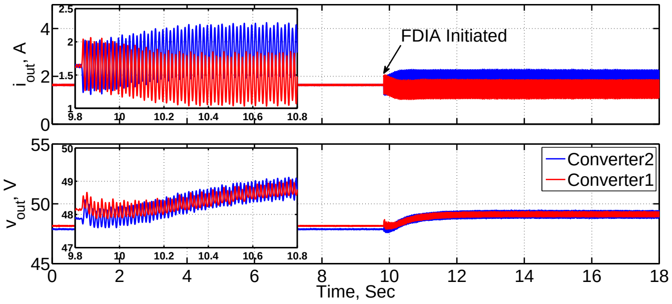


Fig. 8. Experimental data for the constrained FDIA, targeting v_2^{est}.

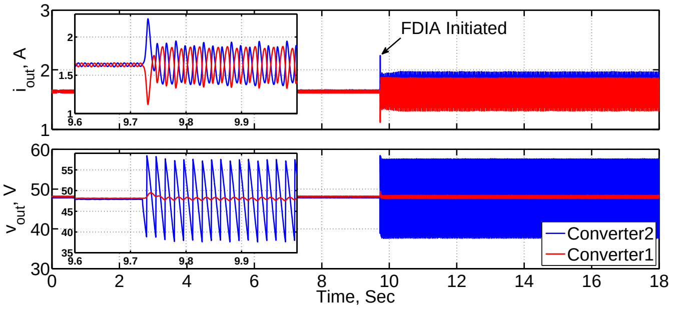


Fig. 11. Experimental data for the constrained FDIA targeting voltage sensor.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2017.2656905, IEEE Transactions on Industrial Informatics

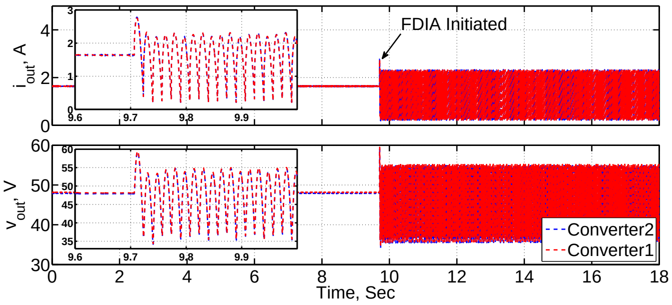


Fig. 12. Experimental data plots for the DC microgrid, under unconstrained FDIA, targeting the entire set of global variables.

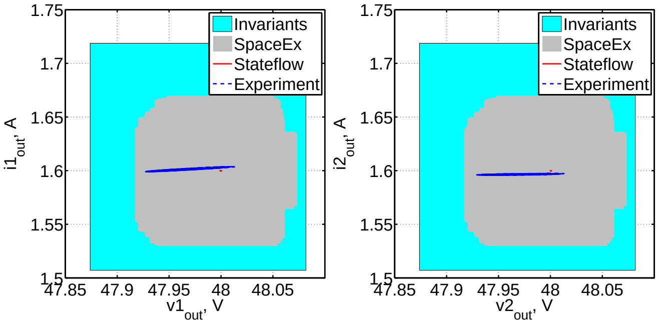


Fig. 13. Phase-portrait comparison of Hynger generated invariants, SpaceEx, SLSF, and experimental results, in the steady state, for DC microgrid under normal conditions (i.e., without FDIA). The experimental and simulation results are contained within the reach sets computed using SpaceEx. Moreover, it is also shown that the SpaceEx reach sets satisfy the invariants.

FDIA scenario are generated automatically using Hynger. This invariant set is then compared with the actual invariants, i.e., invariants under no FDIA scenario to detect intrusion. A comparison of the invariants with and without FDIA scenario is tabulated in Table I. It is evident by comparison that FDIA detection condition mentioned in Definition 3.6, i.e., $\hat{\varphi} \notin \Phi$, is satisfied for the two scenarios, detecting the FDIA.

B. FDIA Mitigation Strategies

Once an FDIA is detected, various mitigation strategies can suppress the effects of the attack. As an example, three possible mitigation strategies are experimentally demonstrated.

1) Physical mitigation strategy: The affected converter may be taken offline after an FDIA is detected. Once the affected converter 2 is disconnected, proper microgrid operation is restored, as shown in Fig. 15.

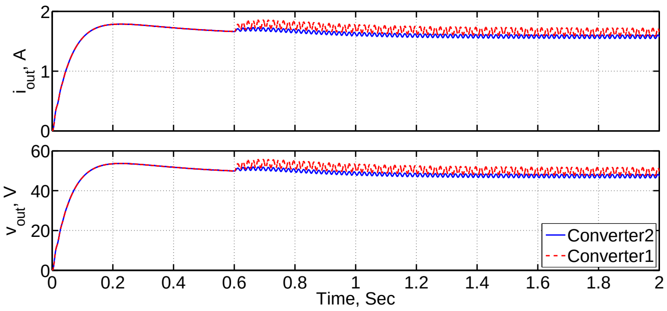


Fig. 14. The SLSF model of DC microgrid is instrumented using Hynger, and the simulation output results for the instrumented model under FDIA scenario are shown demonstrating that the consensus protocol is disturbed. These instrumented traces are passed on to Daikon to generate invariants.

TABLE I
INVARIANTS WITHOUT AND WITH FDIA

Variable	Without FDIA	With FDIA
v_1^{out}	$[47.874, 48.0818]$	$[47.9917, 48.0486]$
v_2^{out}	$[47.8739, 48.0818]$	$[47.9917, 48.5258]$
i_1^{out}	$[1.5071, 1.7187]$	$[1.418, 1.6016]$
i_2^{out}	$[1.5071, 1.7187]$	$[1.5997, 1.6175]$

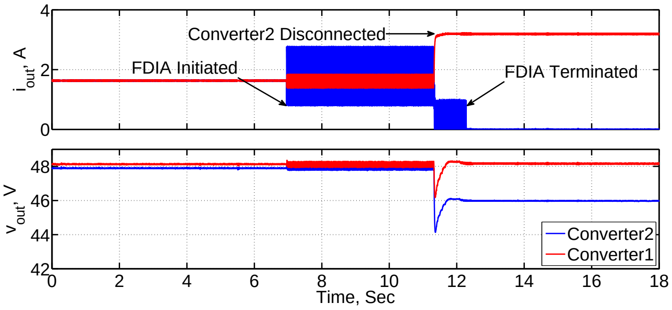


Fig. 15. Experimental data for the constrained FDIA targeting the current sensor of converter 2. The affected converter is disconnected to stabilize the DC microgrid.

2) Communication-based mitigation strategy: The communication link of the effected agent (converter) can be disconnected so that other agents may not be effected. Once the communication link between the affected converter 2 and non-affected converter 1 is disconnected, the output of converter 1 is stabilized, as shown in Fig. 16. The output of converter 2 still remains unstable.

3) Control-based mitigation strategy: One can use a modified control scheme to reduce the effects of FDIA, by augmenting the controller with a false data suppressing mechanism (e.g., filters [39]). As shown in Fig. 17, FDIA is initiated at about 8.5 s, and the modified control scheme is put into action at about 11.97 s to suppress FDIA effects, and the output of the entire DC microgrid is stabilized.

C. Stealthy Attacks with Minimal Weights

The intruder could potentially fabricate an attack vector to bypass the proposed FDIA detection framework, if the changes in candidate invariants, and microgrid operation, are negligible. This is demonstrated through the following two

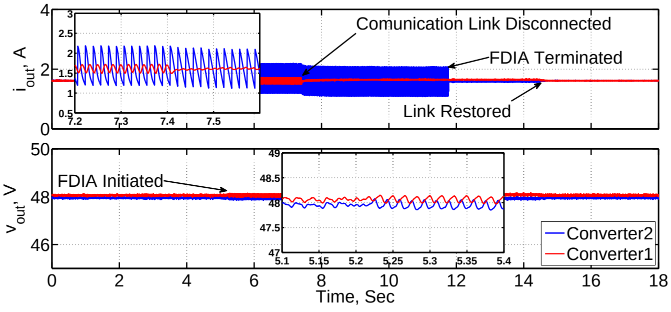


Fig. 16. Experimental data for the constrained FDIA targeting the current sensor of converter 2. The communication link between the affected converter and non-affected converter 1 is disconnected to stabilize converter 1.

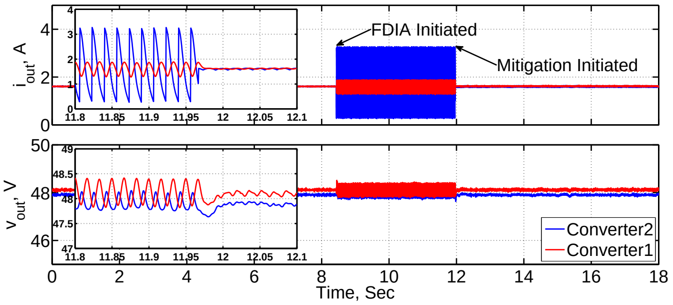


Fig. 17. Experimental data for the constrained FDIA targeting the current sensor of converter 2. As FDIA is detected, the control strategy is augmented with a false data suppression mechanism. It is shown that this controller-based mitigation action has stabilized the entire DC microgrid output.

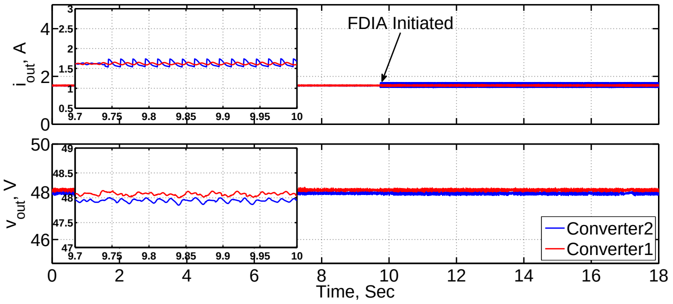


Fig. 18. FDIA, targeting the current sensor of converter 2, with the minimal-weight attack vector that can be detected using this framework.

experiments. First, an attack vector with small weights is designed that can be detected using the proposed framework. The invariants for the output current generated using Hynger are $i_1^{out} = [1.55, 1.77]$ and $i_2^{out} = [1.55, 1.77]$. These invariants are deviated from the corresponding actual invarinats tabulated in Table I, indicating the presence of an FDIA. The negative effects of this FDIA are shown in Fig. 18. It is demonstrated that an FDIA with such minimal destabilizing effects can still be detected using the proposed framework. Next, an attack vector with smaller weights is fabricated to bypass through this FDIA framework. The invariants for the output current generated using Hynger are $i_1^{out} = [1.5071, 1.7187]$ and $i_2^{out} = [1.5071, 1.7188]$ that are comparable with the actual invariants in Table I, hence missing the FDIA. However, the negative effects of this FDIA are negligible, as seen in Fig. 19, as they do not disturb the microgrid operation.

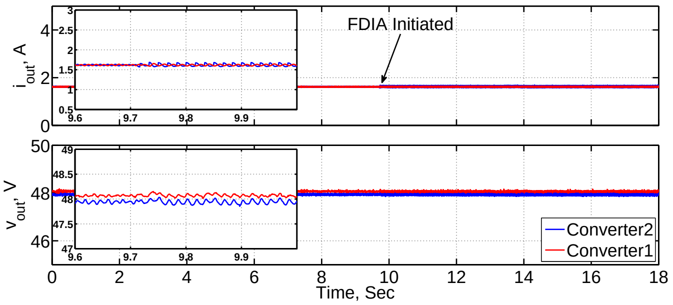


Fig. 19. FDIA, targeting the current sensor of converter 2, with extra minimal-weight attack vector that can bypass the proposed FDIA detection framework. As seen, the effects of FDIA are negligible and do not affect stability.

V. Conclusion

FDIA disturbs the consensus protocols used in the distributed control of cyber-physical DC microgrids. An FDIA detection framework is presented whereby the attack detection problem is formalized as identifying a change in the set of candidate invariants. The candidate invariants are generated using Hynger, that provides an interface between SLSF models and the Daikon tool, which is an invariant inference tool. The hybrid automaton of cyber-physical DC microgrid is presented to obtain the reach sets through reachability analysis. Moreover, the SLSF model of a DC microgrid is also developed to generate the candidate invariants. The actual invariants are obtained after verifying whether the reach sets are contained within the candidate invariants. The candidate invariants generated by Hynger are compared with the actual invariants to successfully detect FDIA.

Appendix

Buck converter parameters are $L = 2.64\,mH, C = 2.2\,mF$, and $F_s = 60\,kHz$. The local loads are $R_1 = R_2 = 30\ \Omega$. The transfer functions are given by:

$$\mathbb{T}_1 = \frac{1}{0.01s+1}, \quad \mathbb{T}_2 = \frac{1}{0.05s+1}. \tag{25}$$

The DC microgrid parameters are: $V_{ref} = [48 \quad 48]^T$, $I_{max} = [4 \quad 4]^T$, $V_{in} = [80 \quad 80]^T, P_{mc} = 0.01, I_{mc} = 1$, $\mathbb{A} = 25\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, $\mathbb{B} = \mathbb{A}, \mathbb{C} = 0.5\mathbb{A}, I = 3\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, P = 0.05\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, K = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$.

References

[1] T. Dragičević, X. Lu, J. C. Vasquez, and J. M. Guerrero, “Dc microgrids - part i: A review of control strategies and stabilization techniques,” *IEEE Transactions on Power Electronics*, vol. 31, no. 7, pp. 4876–4891, July 2016.

[2] ——, “Dc microgrids - part ii: A review of power architectures, applications, and standardization issues,” *IEEE Transactions on Power Electronics*, vol. 31, no. 5, pp. 3528–3549, May 2016.

[3] V. Nasirian, S. Moayedi, A. Davoudi, and F. Lewis, “Distributed cooperative control of dc microgrids,” *IEEE Transactions on Power Electronics*, vol. 30, no. 4, pp. 2288–2303, Apr 2015.

[4] L. Meng, T. Dragičević, J. Roldn-Prez, J. C. Vasquez, and J. M. Guerrero, “Modeling and sensitivity study of consensus algorithm-based distributed hierarchical control for dc microgrids,” *IEEE Transactions on Smart Grid*, vol. 7, no. 3, pp. 1504–1515, May 2016.

[5] X. Lu, X. Yu, J. Lai, J. Guerrero, and H. Zhou, “Distributed secondary voltage and frequency control for islanded microgrids with uncertain communication links,” *IEEE Transactions on Industrial Informatics, doi:10.1109/TII.2016.2603844*, 2016.

[6] F. Luo, Y. Chen, Z. xu, G. Liang, Y. Zheng, and J. Qiu, “Multi-agent based cooperative control framework for microgrids’ energy imbalance,” *IEEE Transactions on Industrial Informatics, doi:10.1109/TII.2016.2591918*, 2016.

[7] Z. Jin, G. Sulligoi, R. Cuzner, L. Meng, J. C. Vasquez, and J. M. Guerrero, “Next-generation shipboard dc power system: Introduction smart grid and dc microgrid technologies into maritime electrical netowrks,” *IEEE Electrification Magazine*, vol. 4, no. 2, pp. 45–57, June 2016.

[8] Q. Li, C. Peng, M. Chen, F. Chen, W. Kang, J. Guerrero, and D. Abbott, “Networked and distributed control method with optimal power dispatch for islanded microgrids,” *IEEE Transactions on Industrial Electronics, doi:10.1109/TIE.2016.2598799*, 2016.

[9] Y. Liu, P. Ning, and M. K. Reiter, “False data injection attacks against state estimation in electric power grids,” *ACM Transactions on Information Systems and Security*, vol. 14, no. 1, pp. 13:1–13:33, Jun. 2011.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TII.2017.2656905, IEEE Transactions on Industrial Informatics

[10] P. Srikantha and D. Kundur, "Denial of service attacks and mitigation for stability in cyber-enabled power grid," in *Proceedings of IEEE Innovative Smart Grid Technologies Conference*, Washington, DC, 2015, pp. 1–5.

[11] X. Zhong, L. Yu, R. Brooks, and G. Venayagamoorthy, "Cyber security in smart dc microgrid operations," in *Proceedings of IEEE 1st International Conference on DC Microgrids*, Atlanta, GA, 2015, pp. 86–91.

[12] Z. Lu, W. Wang, and C. Wang, "Camouflage traffic: Minimizing message delay for smart grid applications under jamming," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 1, pp. 31–44, Jan 2015.

[13] K. Manandhar, X. Cao, F. Hu, and Y. Liu, "Detection of faults and attacks including false data injection attack in smart grid using kalman filter," *IEEE Transactions on Control of Network Systems*, vol. 1, no. 4, pp. 370–379, Dec 2014.

[14] Y. Zou, J. Zhu, X. Wang, and L. Hanzo, "A survey on wireless security: Technical challenges, recent advances, and future trends," *Proceedings of the IEEE*, vol. 104, no. 9, pp. 1727–1765, Sept 2016.

[15] R. B. Bobba, K. M. Rogers, Q. Wang, H. Khurana, K. Nahrstedt, and T. J. Overbye, "Detecting false data injection attacks on dc state estimation," in *Preprints of the First Workshop on Secure Control Systems, CPSWEEK*, Stockholm, Sweden, 2010.

[16] T. T. Kim and H. V. Poor, "Strategic protection against data injection attacks on power grids," *IEEE Transactions on Smart Grid*, vol. 2, no. 2, pp. 326–333, June 2011.

[17] G. Liang, J. Zhao, F. Luo, S. Weller, and Z. Y. Dong, "A review of false data injection attacks against modern power systems," *IEEE Transactions on Smart Grid, doi: 10.1109/TSG.2015.2495133*, 2016.

[18] M. Talebi, C. Li, and Z. Qu, "Enhanced protection against false data injection by dynamically changing information structure of microgrids," in *IEEE 7th Sensor Array and Multichannel Signal Processing Workshop*, Hoboken, NJ, 2012, pp. 393–396.

[19] W. Wang and Z. Lu, "Cyber security in the smart grid: Survey and challenges," *Computer Networks*, vol. 57, no. 5, pp. 1344–1371, Apr 2013.

[20] V. Kounev, D. Tipper, A. A. Yavuz, B. M. Grainger, and G. F. Reed, "A secure communication architecture for distributed microgrid control," *IEEE Transactions on Smart Grid*, vol. 6, no. 5, pp. 2484–2492, Sept 2015.

[21] A. Mazloomzadeh, O. A. Mohammed, and S. Zonouzsaman, "Empirical development of a trusted sensing base for power system infrastructures," *IEEE Transactions on Smart Grid*, vol. 6, no. 5, pp. 2454–2463, Sept 2015.

[22] L. Liu, M. Esmalifalak, Q. Ding, V. A. Emesih, and Z. Han, "Detecting false data injection attacks on power grid by sparse optimization," *IEEE Transactions on Smart Grid*, vol. 5, no. 2, pp. 612–621, Mar 2014.

[23] S. Li, Y. Yilmaz, and X. Wang, "Quickest detection of false data injection attack in wide-area smart grids," *IEEE Transactions on Smart Grid*, vol. 6, no. 6, pp. 2725–2735, Nov 2015.

[24] G. Chaojun, P. Jirutitijaroen, and M. Motani, "Detecting false data injection attacks in ac state estimation," *IEEE Transactions on Smart Grid*, vol. 6, no. 5, pp. 2476–2483, Sep 2015.

[25] D. Rawat and C. Bajracharya, "Detection of false data injection attacks in smart grid communication systems," *IEEE Signal Processing Letters*, vol. 22, no. 10, pp. 1652–1656, Oct 2015.

[26] J. Zhao, G. Zhang, M. La Scala, Z. Dong, C. Chen, and J. Wang, "Short-term state forecasting-aided method for detection of smart grid general false data injection attacks," *IEEE Transactions on Smart Grid*, vol. PP, no. 99, pp. 1–11, Oct 2015.

[27] M. Esmalifalak, L. Liu, N. Nguyen, R. Zheng, and Z. Han, "Detecting stealthy false data injection using machine learning in smart grid," *IEEE Systems Journal*, vol. PP, no. 99, pp. 1–9, Aug 2014.

[28] S. Sridhar and M. Govindarasu, "Model-based attack detection and mitigation for automatic generation control," *IEEE Transactions on Smart Grid*, vol. 5, no. 2, pp. 580–591, Mar 2014.

[29] G. Hug and J. A. Giampapa, "Vulnerability assessment of ac state estimation with respect to false data injection cyber-attacks," *IEEE Transactions on Smart Grid*, vol. 3, no. 3, pp. 1362–1370, Sep 2012.

[30] T. T. Johnson, S. Bak, and S. Drager, "Cyber-physical specification mismatch identification with dynamic analysis," in *Proceedings of 6th International Conference on Cyber-Physical Systems*, Seattle, WA, 2015, pp. 208–217.

[31] M. D. Ernst, J. Cockrell, W. G. Griswold, and D. Notkin, "Dynamically discovering likely program invariants to support program evolution," *IEEE Transactions on Software Engineering*, vol. 27, no. 2, pp. 99–123, Feb 2001.

[32] M. D. Ernst, J. H. Perkins, P. J. Guo, S. McCamant, C. Pacheco, M. S. Tschantz, and C. Xiao, "The Daikon system for dynamic detection of likely invariants," *Science of Computer Programming*, vol. 69, no. 1, pp. 35–45, Dec. 2007.

[33] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, "SpaceEx: Scalable verification of hybrid systems," in *Proceedings of 23rd International Conference on Computer Aided Verification*, Snowbird, UT, 2011, pp. 379–395.

[34] T. Henzinger, "The theory of hybrid automata," in *Proceedings of IEEE Symposium on Logic in Computer Science*, New Brunswick, NJ, 1996, pp. 278–292.

[35] D. Olivares, A. Mehrizi-Sani, A. Etemadi, C. Canizares, R. Iravani, M. Kazerani, A. Hajimiragha, O. Gomis-Bellmunt, M. Saeedifard, R. Palma-Behnke, G. Jimenez-Estevez, and N. Hatziargyriou, "Trends in microgrid control," *IEEE Transactions on Smart Grid*, vol. 5, no. 4, pp. 1905–1919, Jul 2014.

[36] Q. Shafiee, V. Nasirian, J. C. Vasquez, J. M. Guerrero, and A. Davoudi, "A multi-functional fully distributed control framework for ac microgrids," *IEEE Transactions on Smart Grid, doi:10.1109/TSG.2016.2628785*, 2016.

[37] S. Moayedi, V. Nasirian, F. L. Lewis, and A. Davoudi, "Team-oriented load sharing in parallel dc-dc converters," *IEEE Transactions on Industry Applications*, vol. 51, no. 1, pp. 479–490, Jan 2015.

[38] N. Lynch, R. Segala, and F. Vaandrager, "Hybrid I/O automata," *Information and Computation*, vol. 185, no. 1, pp. 105–157, Aug 2003.

[39] F. C. Schweppe and D. B. Rom, "Power system static-state estimation, part i, ii, and iii," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-89, no. 1, pp. 120–135, Jan 1970.

Omar Ali Beg (S'14) received the B.E. and M.S. degrees in electrical engineering from National University of Sciences and Technology, Pakistan. He is presently working toward his Ph.D. degree in electrical engineering at University of Texas at Arlington, TX, USA. He is recipient of the US Air Force Research Laboratory summer research fellowship 2015. His research interests include the formal verification of software-controlled power electronics devices.

Taylor T Johnson (S'05-M'13) received the M.Sc. and Ph.D. degrees in electrical and computer engineering from the University of Illinois at Urbana-Champaign, Urbana, IL, USA, in 2010 and 2013, respectively. He is an Assistant Professor of Electrical Engineering and Computer Science at the Vanderbilt University, Nashville, TN, USA. He received the Air Force Office of Scientific Research Young Investigator Program award in 2016 and the National Science Foundation Computer and Information Science and Engineering Research Initiation Initiative award in 2015. His research interests include developing algorithmic techniques and software tools to improve the reliability of cyber-physical systems.

Ali Davoudi (S'04-M'11-SM'15) received his Ph.D. in electrical and computer engineering from the University of Illinois, Urbana-Champaign, IL, USA, in 2010. He is currently an Associate Professor in the Electrical Engineering Department, University of Texas, Arlington, TX, USA. His research interests include various aspects of modeling and control of power electronics and finite-inertia power systems. Dr. Davoudi is an Associate Editor for IEEE Transactions on Transportation Electrification and IEEE Transactions on Energy Conversion. He has received 2014 Ralph H. Lee Prize paper award from IEEE Transactions on Industry Applications, best paper award from 2015 IEEE International Symposium on Resilient Control Systems, 2014-2015 best paper award from IEEE Transactions on Energy Conversion, and 2016 Prize Paper Award from the IEEE Power and Energy Society.

CyFuzz: A Differential Testing Framework for Cyber-Physical Systems Development Environments

Shafiul Azam Chowdhury, Taylor T. Johnson, and Christoph Csallner

The University of Texas at Arlington, Texas, USA
shafiulazam.chowdhury@mavs.uta.edu
{taylor.johnson,csallner}@uta.edu

Abstract. Designing complex systems using graphical models in sophisticated development environments is becoming de-facto engineering practice in the cyber-physical system (CPS) domain. Development environments thrive to eliminate bugs or undefined behaviors in themselves. Formal techniques, while promising, do not yet scale to verifying entire industrial CPS tool chains. A practical alternative, automated random testing, has recently found bugs in CPS tool chain components. In this work we identify problematic components in the Simulink modeling environment, by studying publicly available bug reports. Our main contribution is CyFuzz, the first differential testing framework to find bugs in arbitrary CPS development environments. Our automated model generator does not require a formal specification of the modeling language. We present prototype implementation for testing Simulink, which found interesting issues and reproduced one bug which MathWorks fixed in subsequent product releases. We are working on implementing a full-fledged generator with sophisticated model-creation capabilities.

Keywords: Differential testing, cyber-physical systems, model-based design, Simulink

1 Introduction

Widely used cyber-physical system (CPS) development tool chains are complex software systems that typically consist of millions of lines of code [1]. For example, the popular MathWorks Simulink tool chain contains model-based design tools (in which *models* in various expressive modeling languages are used to describe the overall system under control [2]), simulators, compilers, and automated code generators. Like any complex piece of code, CPS tool chains may contain bugs and such bugs may lead to severe CPS defects.

The vast majority of resources in the CPS design and development phases are devoted to ensure that systems meet their specifications [3, 4]. In spite of having sophisticated design validation and verification approaches (model checking, automated test case generation, hardware-in-the-loop and software-in-the-loop testing etc.), we see frequent safety recalls of products and systems among industries, due to CPS bugs [5–7].

Since many CPSs operate in safety-critical environments and have strict correctness and reliability requirements [8], it would be ideal for CPS development tools to not have bugs or unintended behaviors. However, this is not generally true as demonstrated by recent *random testing* projects finding bugs in a static analysis tool (Frama-C) [9] and in popular C compilers (GCC and LLVM) [10], which are widely used in CPS model-based design.

It would be extremely expensive or possibly even practically infeasible to formally verify entire CPS tool chains. In addition to their sheer size in terms of lines of code, a maybe more significant hurdle is the lack of a complete and up to date formal specification of the CPS tool chain semantics, which may be due to their complexity and rapid release cycles [1, 11].

Instead of formally verifying the absence of bugs in all CPS tool chain execution paths, we revert to showing the presence of bugs on individual paths (aka testing), which can still be a major contributor to software quality [12]. *Differential testing* or *fuzzing*, a form of random testing, mechanically generates random test inputs and presents them to comparable variations of a software [12]. The results are then compared and any variation from the majority (if one exists) likely indicates a bug [13]. This scheme has been effective at finding bugs in compilers and interpreters of traditional programming languages. As an example, various fuzzing schemes have collectively found over 1,000 bugs in widely used compilation tools such as GCC [10, 11, 14].

While compiler testing is promising, when testing CPS tool chains we face additional challenges beyond what is covered by testing compilers of traditional programming languages (such as Csmith creating C programs), since CPS modeling languages differ significantly from traditional programming languages. A key difference is that the complete semantics of widely used commercial modeling languages (e.g., MathWorks Simulink and Stateflow [15]) are not publicly available [1, 16, 17]. Moreover, modeling language semantics often depend on subtle details, such as two-dimensional layout information, internal model component settings, and the particular interpretation algorithm of simulators [1]. Finally, random generation of test cases for CPS development environments has to address a combination of programming paradigms (e.g., both graphical, data-flow language and textual imperative programming language in the same model), which is rare in traditional compiler testing.

Since existing testing and verification techniques are not sufficient for ensuring the reliability of CPS tool chains, we propose CyFuzz: a novel conceptual differential testing framework for testing arbitrary CPS development environments. We use the term *system under test (SUT)* to refer to the CPS tool chain being tested. CyFuzz has a *random model generator* which automatically generates random CPS models the SUT may simulate or compile to embedded native code. CyFuzz's *comparison framework* component then detects dissimilarity (if it exists) in the results obtained by *executing* (or, *simulating*) the generated model, by varying components of the SUT.

We also present an implementation for testing the Simulink environment, which is widely used in CPS industries for model-based design of dynamic and embedded systems [18, 19]. Although our current prototype implementation targets Simulink, the described conceptual framework is not tool specific and should thus be applicable to related CPS tool chains, such as NI's LabVIEW [20].

To the best of our knowledge, CyFuzz is the first differential testing framework for fuzzing CPS tool chains. To address the problem of missing formal semantics during model generation, we follow a simple, feedback-driven model generation approach that iteratively fixes generated models according to the SUT's error descriptions. To summarize, this paper makes the following contributions:

- To understand the types of Simulink bugs that affect users, we first analyze a subset of the publicly available Simulink bug reports (Section 3).
- We present CyFuzz, a conceptual framework for (1) generating random but valid models for a CPS modeling language, (2) simulating the generated models on alternative CPS tool chain configurations, and (3) comparing the simulation results (Section 4). We then describe interesting implementation details and challenges of our prototype implementation for Simulink (Section 5).
- We report on our experience of running our prototype tool on various Simulink configurations (Section 6), identifying comparison errors and semi-independently reproducing a confirmed bug in Simulink's `Rapid Accelerator` mode.

2 Background: Model-based CPS Design and Simulink

This section provides necessary background information on model-based development. We define the terms used for explaining a conceptual differential testing framework and subsequently relate them with Simulink.

2.1 CPS Model Elements

The following concepts and terms are applicable to many CPS modeling languages (including Simulink). A *model*, also known as a *block-diagram*, is a mathematical representation of some CPS [18]. Designing a diagram starts with choosing elementary elements called blocks. Each *block* represents a component of the CPS and may have *input* and *output* ports. An input port accepts data on which the block performs some operation. An output port passes data to other input ports using *connections.* An output port can be connected to more than one input port while the opposite is not true in general. A Block may have *parameters*, which are configurable values that influence the block's behavior. Somewhat similar to a programming language's standard libraries, a CPS tool chain typically provides *block libraries*, where each library consists of a set of predefined blocks.

Since hierarchical models are commonly found in industry, CyFuzz supports generating such models as well. This can be achieved by grouping some blocks of a model together and replacing them by a new block which We call a *child*, whereas the original model is called *parent*.

When simulating, the SUT numerically solves the mathematical formulas represented by the model [18]. Simulation is usually time bound and at each *step* of the simulation, a *solver* calculates the blocks' outputs. We use the term *signal* to mean output of a block's port at a particular simulation step.

The very first phase of the simulation process is *compiling* the model. This stage also looks for incorrectly generated models and raises failures for syntactical model errors, such as data type mismatches between connected output and input ports. If an error is found in the compilation phase, the SUT does not attempt simulating the model. After successful simulation, *code generators* can generate native code, which may be deployed in target hardware [1].

2.2 Example CPS Development Environment: Simulink

While our conceptual framework uses the above terms, they also apply directly in the context of Simulink [21]. Besides having a wide selection of built-in blocks, Simulink allows integrating native code (e.g., Matlab or C code) in a model via Simulink's `S-function` interface, which lets users create custom blocks for use in their models. Simulink's `Subsystem` and `Model referencing` features enable hierarchical models.

Simulink has three simulation modes. In `Normal` mode, Simulink does not generate code for blocks, whereas it generates native code for certain blocks in the `Accelerator` mode. Unlike in these two modes, the `Rapid Accelerator` mode further creates for the model a standalone executable. To capture simulation results we use Simulink's `Signal Logging` functionality as we found implementing it quite feasible. However, for cases where the approach is not applicable (see [21]), we use Simulink's `sim` api to record simulation data.

3 Study of Existing Bugs: Incorrect Code Generation

To understand the types of bugs Simulink users have found and care about, we performed a study on the publicly available bug reports from the MathWorks website[1]. We identified commonalities in bug reports, which we call *classification factors*. We limited our study to bug reports found via the search query *incorrect code generation*, as earlier studies have identified code generation as vulnerable [1, 22].

We investigated bug reports affecting Matlab/Simulink version 2015a as we were using it in our experiments. As of February 17, 2016, there were 50 such bug reports, among which 47 have been fixed in subsequent releases of the products. Table 1 summarizes the findings. Our complete study data are available at: `http://bit.ly/simstudy`

Table 1 shows only those classification factors that affect at least 20% of all the bug reports that we have studied. We use insights obtained from the

[1] Available: `http://www.mathworks.com/support/bugreports/`

Table 1. Study of publicly available Simulink bug reports. The right column denotes the percentage of bug reports affected by a the given classification factor. Each bug report may be classified under multiple factors.

Classification factor	Bugs [%]
Reproducing the bug requires a code generator to generate code	60
Reproducing the bug requires specific block parameter values and/or port or function argument values and data-types	56
Reproducing the bug requires comparing simulation-result and generated code's output	54
Reproducing the bug requires connecting the blocks in a particular way	36
Reproducing the bug requires specific model configuration settings	32
Reproducing the bug requires hierarchical models	24
Reproducing the bug requires built-in Matlab functions	20

study in our CyFuzz prototype implementation. For example, many of the bug reports (54%) are related to simulation result and generated code execution output mismatch. Thus, differential testing (e.g., by comparing simulation and code execution) seems like a good fit for finding bugs in CPS tool chains. Further insight that is reflected in our tool is that it is worth exploring the large space of possible block connections (36% of bug reports) e.g., via random block and connection generation. Other insights we want to use in the future are to incorporate random block parameter values and port data-types (56%) and model configurations (32%).

4 Differential Testing of CPS Development Tool Chains

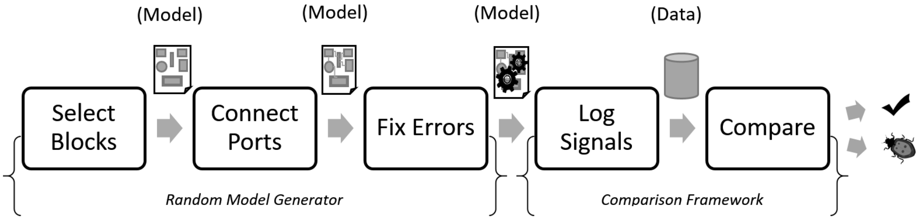


Fig. 1. Overview of the differential testing framework. The first three *phases* correspond to the random model generator, while the rest belongs to the comparison framework.

At a high level we can break our objective into two sub goals: creating a *random model generator* and defining a *comparison framework*. We first present a theory applicable to a conceptual CPS framework in this section. Fig. 1 provides a schematic overview of CyFuzz's processing *phases*. The first three phases belong to the random model generator, and the remaining two constitute the comparison framework. The first two phases create a random model (which may violate Simulink's model construction rules). The third phase fixes many of these errors, such that the model passes the SUT's type checkers and the SUT can simulate it. If it succeeds it passes the model to the fourth phase to simulate the model in various SUT configurations and to record results. The final phase detects any dissimilarities in the collected data, which we call *comparison error* bugs.

4.1 Conceptual Random Model Generator

Following are details on the generator's three phases.

Listing 1.1. Select Blocks phase of the conceptual random model generator.

```
method select_blocks ($n$, $block\_libraries$):
    /* Choose $n$ blocks from the given $block\_libraries$, place the blocks
       in a new model, configure the blocks, and return the model. */
    $m$ = create_empty_model() // New, empty model
    $blocks$ = choose_blocks($n$, $block\_libraries$) // $N$ from $block\_libraries$
    for each block $b$ in $blocks$:
        place_block_in_model(m, $b$)
        configure_block($b$, $n$, $block\_libraries$)
    return $m$
```

Select Blocks. Listing 1.1 summarizes this phase, which selects, places, and configures the model's blocks. The generator has a list of block libraries and for each library a predetermined weight. Using the weights, the *choose_blocks* method selects n random blocks. The value n can be fixed or randomly selected from a range. On a newly created model the generator next places each of these blocks using the *place_block_in_model* method. For creating inputs, CyFuzz selects various kinds of blocks, to, for example, provide random inputs to the model.

The *configure_block* method selects block parameter values and satisfies some block constraints (e.g., by choosing blocks required for placing a certain block). For creating hierarchical models, a child model is considered as a regular block in the parent model and is passed as a parameter to *configure_block*, which calls *select_blocks* to create a new child model. Here n is equal to the parent model, but *block_libraries* may not be the same (e.g., certain blocks are not allowed in some Simulink child models).

Connect Ports. The second phase follows a simple approach to maximize the number of ports connected. CyFuzz arbitrarily chooses an output and an input port from the model's blocks, prioritizing unconnected ports. It then connects them and continues the process until all input ports are connected. Consequently, some output ports may be left unconnected.

Listing 1.2. fix_errors tries to fix the model errors that the *simulate* method raises; p is a SUT configuration; t denotes a timeout value.

```
method fix_errors ($m$, $p$, $attempt\_limit$, $t$):
    for $i$ = 1 to $attempt\_limit$:
        $< r^p_{status}, r^p_{data}, errors >$ = simulate($m$, $p$, $t$)
        if $r^p_{status}$ is $error$:
            if fix_model($m$, $errors$) is $false$:
                return $< r^p_{status}, r^p_{data}, errors >$
        else:
            return $< r^p_{status}, r^p_{data}, errors >$
    return simulate($m$, $p$, $t$)
```

Fix Errors. Because of their simplicity, CyFuzz's first two phases may generate invalid models that cannot be simulated successfully. The third phase tries to fix these errors. Listing 1.2 outlines the approach. It uses method *simulate* to simulate model m up to time $t \in \mathbb{R}^+$ (in milliseconds) using SUT configuration p.

The *simulate* output is a 3-tuple, where r^p_{status} is one of $success$, $error$, or $timed - out$. Note that first step of simulation is compiling the model (see Section 2). If m has errors, *simulate* will abort compilation, storing error-related diagnostic information in $errors$. r^p_{data} contains simulation results (time series data of the model's blocks' outputs) if $r^p_{status} = success$.

At this point we assume that the error messages are informative enough to drive the generator. For example, Simulink satisfies this assumption. Using $errors$, fix_model tries to fix the errors by changing the model. As it changes the model this phase may introduce new errors. We try to address such secondary errors in subsequent loop iterations in Listing 1.2, up to a configurable number $attempt\_limit$. While this approach is clearly an imperfect heuristic, it has worked relatively well in our preliminary experience (as, e.g., is indicated by the low error rate in Table 2).

4.2 Conceptual Comparison Framework

Here we explore simulating a randomly generated model varying SUT-specific configuration options of a CPS tool chain, and thus testing it in two phases.

Log Signals. If simulation was successful in the Fix Errors phase, CyFuzz simulates the model varying configurations of the SUT in this phase; let P be such a set of configurations. Using the *simulate* method introduced in Section 4.1, for each $p \in P$ we calculate $< r^p_{status}, r^p_{data}, errors >= simulate(m, p, t)$ for a model m and add r^p_{data} to a set d only if $r^p_{status} = success$. We pass d to next phase of the framework. r^p_{data} should contain time series data of the output ports of the model's blocks at all available simulation steps. In the next phase, however, we use only the values recorded at the last simulation step; we leave comparing signal values at other simulation steps as future task.

Compare. In its last phase, CyFuzz compares the recorded simulation results d obtained in the previous phase using method *compare* (Listing 1.4). It uses method *retrieve*, which returns the signal value of a particular block's particular port at a given time instance. If the value is not available (e.g., blocks that do not have output ports do not participate in signal logging), it returns the special value *Nil*. *compare* also uses method *latest_time* which returns the time of the last simulation step for a given block's particular port. If no data is available, it returns *Nil*.

Listing 1.3. Determining equivalence via tolerance limit ϵ.

```
method equiv ($p$, $q$):
  if $p$ and $q$ are $Nil$: // Missing both data points
    return $true$
  if $p$ or $q$ is $Nil$: // Missing one data point
    return $false$
  return $|p - q| < \epsilon$
```

Listing 1.4. This method compares two execution results (of model m) taken as first two arguments and throws errors if it finds a dissimilarity.

```
method compare ($r^p_{data}$, $r^q_{data}$, $m$):
    for each block $b$ of the model $m$:
        for each output port $y$ of the block $b$:
            $t_p$ = latest_time($r^p_{data}$, $b$, $y$)
            $t_q$ = latest_time($r^q_{data}$, $b$, $y$)
            if $equiv(t_p, t_q)$ is $false$:
                throw ‘‘Time Mismatch’’ error
            else if $t_p \neq Nil$:
                if $equiv(retrieve(r^p_{data}, b, y, t_p), retrieve(r^q_{data}, b, y, t_q))$ is $false$:
                    throw ‘‘Data Mismatch’’ error
```

Now, taking two elements from d at a time we form all possible pairs (r^p_{data}, r^q_{data}) where $p \neq q$ and apply method *compare* on them. As comparing floating-point numbers using straight equality checking is problematic [1, 23], *eqiv* (Listing 1.3) method uses a tolerance limit to determine floating-point equivalence. If *compare* reports an error, we mark m as a *comparison error* for p, q and submit it to manual inspection.

5 CyFuzz Prototype Implementation for Simulink

We have developed a prototype implementation of CyFuzz mostly in Matlab. The tool continuously generates one Simulink model at a time and then passes it to the comparison framework. Source code, implementation and usage details, sample generated models, and detailed experiment results are available at: `https://github.com/verivital/slsf_randgen`.

Selecting and Configuring Blocks. Simulink itself has over 15 built-in libraries. MathWorks also offers `toolboxes`, which add to Simulink additional libraries. To date we have included in our experiments blocks from only four of these libraries, `Sources`, `Sinks`, `Discrete`, and `Concrete`. We use default parameter values for configuring most blocks. However, some Simulink blocks do not allow placing multiple instances of the same block with the same default value in a model. For these blocks we randomly choose parameter values.

Generating Hierarchical Models. Since hierarchical models are very popular among Simulink users, our prototype can generate them. Currently, the generator uses `Model referencing` and `For each subsystems` blocks to create hierarchical models. CyFuzz generates model hierarchies up to a configurable depth. In doing so it places and configures related blocks. For example, CyFuzz automatically puts input (output) related blocks in a new child model which are used to accept (return) data from (to) the parent model. The number of blocks for the top-level and child models are chosen randomly from user-provided ranges.

Fix Errors Phase. We utilize Matlab's exception handling mechanism to learn what prevented successful compilation of the model. Some information (e.g., the error type) can be directly collected from the exception. Collecting other important information, such as the actual problematic block, can be nontrivial. For example, for algebraic loop errors sometimes CyFuzz has to identify other blocks (e.g., a parent block) to fix the problem. As another example, the current CyFuzz version does not attempt to know the data types of the ports in the Connect Ports phase. Rather, it collects such information when compiling the model using diagnostic information returned by the SUT.

Models with Random Native Code. To facilitate blocks with custom behavior, Simulink allows placing native code (C, Matlab etc.) directly in models. To generate such blocks we leverage Csmith, which generates random C programs [10]. We designed simple Simulink blocks using Matlab's `S-function` interface that use random code generated by a customized version of Csmith. Our customized version is capable of generating many different C functions that can be called from various simulation steps. We looked for both crash errors and "wrong code errors" (similar to our comparison error). However, this is not fully integrated with CyFuzz yet.

The Comparison Framework. CyFuzz starts with varying simulation modes (see Section 2.2). and compiler optimization levels. For instance, "`Normal` mode", "`Accelerator` mode; optimization on", and "`Rapid Accelerator`; optimization off" are options to vary. Varying compilers, code generators, solver-specific settings, and other possible SUT configuration options are future work.

6 Experience with CyFuzz

Here we analyze our prototype implementation based on experimental results.

6.1 Research Questions (RQ), Experimental Setup, and Results

Throughout this work we explore the following research questions.

RQ1 Is the random model generator effective? Which portion of the generated models can the SUT compile and simulate within a given time bound?
RQ2 Using the generated models, can the comparison framework effectively find bugs (comparison errors or crashes) in the SUT ?
RQ3 What is the runtime of each of CyFuzz's stages? Does the generator scale with the generated model's number of blocks?

To answer these questions we conducted experiments using Matlab `2015a` on Ubuntu 14.10 and varied simulation mode (`Normal` vs. `Accelerator`) and optimizer (on vs. off) for the later mode. For the *fix_errors* method (Listing 1.2) we chose *attempt_limit* 10 and *timeout* 12. For choosing blocks we used a traditional $O(n)$ implementation of the *fitness proportion selection* algorithm [24]. We have not included in these experiments hierarchical models or custom blocks.

Table 2. Each row represents a separate experiment. Columns 3–6 is the percentage of blocks selected per library (e.g., experiment A chose 80% of the blocks from the `Discrete` library). *Error* denotes the number of models that failed to simulate. *Timed-out* denotes the models that did not complete simulation within the time bound.

Exp.	Total	`Discrete`	`Concrete`	`Source`	`Sink`	error	timed-out	Confirmed
Label	Models	[%]	[%]	[%]	[%]	[%]	[%]	Bugs [%]
A	1172	80	0	10	10	9.73	0.60	0
B	1095	43	37	10	10	1.74	7.03	0
C	1449	0	80	10	10	12.01	8.63	0

Table 3. More information on experiments from Table 2. Columns 3-7 denotes the time taken by the five phases of CyFuzz. *Runtime* denotes the average time CyFuzz spent for a model.

Exp.	Blocks/	Select	Connect	Fix	Log	Compare	Runtime
Label	Model	Blocks [%]	ports [%]	Errors [%]	Signals [%]	[%]	[sec]
A	35.00	7.85	0.64	16.00	74.55	0.96	40.37
B	34.96	6.06	0.39	16.06	76.86	0.63	51.87
C	35.05	8.09	0.51	11.02	79.58	0.80	42.51

Effectively Creating Random Models (RQ 1). As the experimental results in Table 2 suggest, our tool can generate many models that Simulink can successfully simulate. For each row in the table we have a low error and timed-out rate. This high success rate is crucial for the framework as it only uses such valid models in the tool's later comparison framework phases. We also observed that the number of errors and timed-out models varied with the selected block libraries, but we have not yet analyzed the reasons of these variations.

Effectiveness of Comparison Framework (RQ 2). We have not found new bugs yet, however, our framework reproduced an existing bug and found interesting cases (see Section 6.2).

Runtime Analysis (RQ 3). The Select Blocks algorithm of Listing 1.1 has runtime $O(n)$, n being the number of blocks in the model and using an $O(1)$ block selection algorithm. The random model generator scales linearly with the number of blocks. But as the number of blocks grows, the number of timed-out models and errors also grow. A preliminary analysis suggests that there are relatively few distinct error causes. We group errors by their causes and fixing one cause dramatically increased the overall number of successfully executed models.

Table 3 indicates that the Log Signals phase uses most of the runtime. This result is not surprising, as in this phase the SUT simulates the model, generates and executes code, and logs the data, all of which are time consuming tasks.

Using Native Code/Custom Blocks. In separate experiments we used a fixed Simulink model with a custom block created using `S-Function`. We repeatedly generated random C code using a customized version of Csmith and plugged this code in the `S-function`, which effectively ran the code once we simulated the model. We used different optimizer settings for GCC when compiling and were able to reproduce crash and "wrong code" bugs of GCC 4.4.3. This shows that incorporating Csmith in our framework is promising. However, more work is needed to fully utilize Csmith-generated programs and create sophisticated Simulink blocks using them. One limitation is that floating-point support in Csmith is currently still basic and can only be used for detecting crash-bugs.

6.2 Interesting Comparison Framework Findings

Following are two interesting findings of our experiments, including one independently rediscovered confirmed Simulink bug.

Comparison Error for Models with Algebraic Loops. In our experiments we noticed comparison errors for some models where Simulink solved algebraic loops. Investigating further we noticed that when Simulink solves an algebraic loop it is not confident of its correctness [21]. For this, we did not classify this case as a bug. CyFuzz now eliminates algebraic loops altogether rather than relying on Simulink to solve them. We note that one can use our tool to opportunistically discover such inaccuracies for models with algebraic loops and decide whether to accept Simulink's solution for solving the loops.

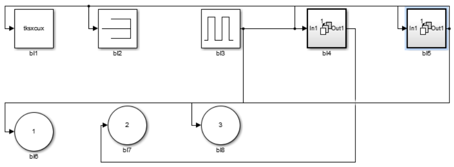


Fig. 2. Screen-shot of generated top-level Simulink model which reproduced a bug

Bug in Simulink's `Rapid Accelerator` Mode. In separate experiments with hierarchical models, we noticed that for a model (see Fig. 2) values of a Simulink `Outport` block are significantly different in `Normal` and `Rapid Accelerator` mode. This was detected automatically by our comparison framework. After submitting a bug report MathWorks confirmed that the case was already identified as a bug and they fixed it for later versions.

7 Future Work and Discussion

Our ultimate goal is to provide a full-fledged fuzz-testing framework for Simulink. Our work on CyFuzz and our prototype implementation for Simulink are thus both ongoing. Following is a sample of the opportunities for improvement.

The current prototype implementation has several limitations. Currently, the tool chooses blocks from only four built-in libraries. Incorporating additional libraries will increase the expressiveness of generated models and thus its potential for finding bugs. Also, we plan on integrating custom blocks developed using native code and perform experiments we were not able to conduct yet.

The comparison framework implementation is also not free from shortcomings. So far, we have only used various simulation modes and compiler optimization levels. However, we are interested in adding more variations (e.g. those listed in Section 5). Finally, CyFuzz should compare signals in multiple simulation steps, since it was also found effective in previous work [25].

8 Related Work

The following focuses on the most closely related work not covered by the introduction section. Existing approaches for CPS testing mostly aim at generating test cases for existing models (e.g., [26, 18]) and do not target testing of CPS tool chains. *Code generator testing* ([1, 27]) only target a relatively small component of the CPS tool chain but not an entire CPS tool chain.

Most of the compiler fuzzers perform random walks over a context-free grammar, thus mainly focusing on generating syntactically valid [14] and well typed programs in imperative languages [28, 10, 11, 29]. None of the works target dataflow languages like Simulink. We find Csmith most related to our work, which is state-of-the-art C compiler fuzzer. Csmith leverages the well-published C99 standard and can be used to test only a component of entire CPS tool chain [10]. Our test generation and comparison techniques differ fundamentally from Csmith. Conceptually, CPS tool chain fuzzing is a super-set of the schemes presented in Csmith. CPS tool chains typically contain a C compiler; thus CyFuzz leverages Csmith as a component.

Earlier work includes a differential testing based runtime verification framework, leveraging a random hybrid automata generator [30, 25]. Other works attack code generators used in CPS tool chain. Stürmer et al. generate model taking specification of a code generator's optimization rules in graph grammar [1]. But such specifications for code generators might not be available and white-box testing in parts is undesirable [31]. Sampath et al. propose testing *model-processing tools* taking semantic meta-model of Stateflow (a Simulink component) [31]. But the approach does not scale and the complete specifications it needs are not available. In contrast, we propose the first fuzz-testing framework to test arbitrary CPS tool chains based on feasible model generation.

Many CPS model verification and safety checking approaches have been proposed [8, 32]. Recent work verifies existing SL/Stateflow (SL/SF) models by generating test inputs for these models [18, 19]. Alur et al. analyze generated symbolic traces of a SL/SF model, and combine simulation and symbolic analysis for improving coverage of given SL/SF models [33]. The *Simulink Code Inspector* compares generated code for a given model based on structural equivalence and traceability [21]. However none of these approaches describe random generation of Simulink models for fuzzing the CPS tool chain.

9 Conclusions

This work addresses the CPS tool chain quality problem using a differential testing scheme. Existing work either does not test CPS development tool chains or only tests small subsets. As CPS tool chains are actively developed and released, formal specification based test generation schemes are not suitable for fuzzing CPS tool chains. Rather, our approach follows a simple model generation strategy applicable to arbitrary CPS modeling languages. Starting with a random and possibly erroneous model, our generator fixes various errors in the model using diagnostic information returned by the system under test. In our experiments a high portion of the generated models could thus be executed without errors.

We also define techniques to find bugs in CPS tool chains based on simulation result comparison. The approach is effective as our prototype implementation for Simulink found interesting cases and one bug. Although our model generator is scalable and fully automatic, more work is needed to systematically search the huge space of possible data-flow models and generate those models that are likely to find bugs in modern CPS development environments.

Acknowledgments. This material is based upon work supported by the National Science Foundation under Grants No. 1117369, 1464311, and 1527398, by Air Force Office of Scientific Research (AFOSR) contract numbers FA9550-15-1-0258 and FA9550-16-1-0246, and by Air Force Research Lab (AFRL) contract number FA8750-15-1-0105. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of AFRL, AFOSR, or NSF.

References

1. Stürmer, I., Conrad, M., Dörr, H., Pepper, P.: Systematic testing of model-based code generators. IEEE Transactions on Software Engineering (TSE) **33**(9) (September 2007) 622–634
2. Lee, E.A., Seshia, S.A.: Introduction to Embedded Systems: A Cyber-physical Systems Approach. First edn. `http://LeeSeshia.org` (2011)
3. Beizer, B.: Software testing techniques. Second edn. Van Nostrand Reinhold (June 1990)
4. U.S. National Institute of Standards and Technology (NIST): The economic impacts of inadequate infrastructure for software testing: Planning report 02-3 (May 2002)
5. U.S. Consumer Product Safety Commission (CPSC): Recall 11-702: Fire alarm control panels recalled by fire-lite alarms due to alert failure. http://www.cpsc.gov/en/Recalls/2011/Fire-Alarm-Control-Panels-Recalled-by-Fire-Lite-Alarms-Due-to-Alert-Failure (October 2010)
6. U.S. National Highway Traffic Safety Administration (NHTSA): Defect Information Report 14V-053. `http://www-odi.nhtsa.dot.gov/acms/cs/jaxrs/download/doc/UCM450071/RCDNN-14V053-0945.pdf` (February 2014)
7. Alemzadeh, H., Iyer, R.K., Kalbarczyk, Z., Raman, J.: Analysis of safety-critical computer failures in medical devices. IEEE Security Privacy **11**(4) (July 2013) 14–26
8. Johnson, T.T., Bak, S., Drager, S.: Cyber-physical specification mismatch identification with dynamic analysis. In: Proc. ACM/IEEE Sixth International Conference on Cyber-Physical Systems (ICCPS), ACM (April 2015) 208–217
9. Cuoq, P., Monate, B., Pacalet, A., Prevosto, V., Regehr, J., Yakobowski, B., Yang, X.: Testing static analyzers with randomly generated programs. In: Proc. 4th NASA Formal Methods Symposium (NFM), Springer (April 2012) 120–125
10. Yang, X., Chen, Y., Eide, E., Regehr, J.: Finding and understanding bugs in C compilers. In: Proc. 32nd ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), ACM (June 2011) 283–294
11. Dewey, K., Roesch, J., Hardekopf, B.: Fuzzing the Rust typechecker using CLP (T). In: Proc. 30th IEEE/ACM International Conference on Automated Software Engineering (ASE), IEEE (2015) 482–493
12. McKeeman, W.M.: Differential testing for software. Digital Technical Journal **10**(1) (1998) 100–107
13. Lidbury, C., Lascu, A., Chong, N., Donaldson, A.F.: Many-core compiler fuzzing. In: Proc. 36th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), ACM (June 2015) 65–76
14. Holler, C., Herzig, K., Zeller, A.: Fuzzing with code fragments. In: Proc. 21th USENIX Security Symposium, USENIX Association (August 2012) 445–458
15. The MathWorks Inc.: Products and services. `http://www.mathworks.com/products/` (2016)

16. Hamon, G., Rushby, J.: An operational semantics for Stateflow. International Journal on Software Tools for Technology Transfer **9**(5) (2007) 447–456
17. Bouissou, O., Chapoutot, A.: An operational semantics for Simulink's simulation engine. In: Proc. 13th ACM SIGPLAN/SIGBED International Conference on Languages, Compilers, Tools and Theory for Embedded Systems (LCTES), ACM (June 2012) 129–138
18. Matinnejad, R., Nejati, S., Briand, L.C., Bruckmann, T.: SimCoTest: A test suite generation tool for Simulink/Stateflow controllers. In: Proc. 38th International Conference on Software Engineering, (ICSE), ACM (May 2016) 585–588
19. Sridhar, A., Srinivasulu, D., Mohapatra, D.P.: Model-based test-case generation for Simulink/Stateflow using dependency graph approach. In: Proc. 3rd IEEE International Advance Computing Conference (IACC). (February 2013) 1414–1419
20. National Instruments: Labview system design software. `http://www.ni.com/labview/` (2016)
21. The MathWorks Inc.: Simulation documentation. `http://www.mathworks.com/help/simulink/` (2016)
22. Rajeev, A.C., Sampath, P., Shashidhar, K.C., Ramesh, S.: CoGenTe: A tool for code generator testing. In: Proc. 25th IEEE/ACM International Conference on Automated Software Engineering (ASE), ACM (September 2010) 349–350
23. Goldberg, D.: What every computer scientist should know about floating-point arithmetic. ACM Comput. Surv. **23**(1) (1991) 5–48
24. Goldberg, D.E.: Genetic algorithms in search, optimization and machine learning. First edn. Addison-Wesley (1989)
25. Nguyen, L.V., Schilling, C., Bogomolov, S., Johnson, T.T.: Runtime verification of model-based development environments. In: Proc. 15th International Conference on Runtime Verification (RV). (September 2015)
26. Girard, A., Julius, A.A., Pappas, G.J.: Approximate simulation relations for hybrid systems. Discrete Event Dynamic Systems **18**(2) (2008) 163–179
27. Stürmer, I., Conrad, M.: Test suite design for code generation tools. In: Proc. 18th IEEE International Conference on Automated Software Engineering (ASE). (October 2003) 286–290
28. Csallner, C., Smaragdakis, Y.: JCrasher: An automatic robustness tester for Java. Software—Practice & Experience **34**(11) (September 2004) 1025–1050
29. Hussain, I., Csallner, C., Grechanik, M., Xie, Q., Park, S., Taneja, K., Hossain, B.M.: Rugrat: Evaluating program analysis and testing tools and compilers with large generated random benchmark applications. Software—Practice & Experience **46**(3) (March 2016) 405–431
30. Nguyen, L.V., Schilling, C., Bogomolov, S., Johnson, T.T.: HyRG: A random generation tool for affine hybrid automata. In: Proc. 18th International Conference on Hybrid Systems: Computation and Control (HSCC), ACM (April 2015) 289–290
31. Sampath, P., Rajeev, A.C., Ramesh, S., Shashidhar, K.C.: Testing model-processing tools for embedded systems. In: Proc. 13th IEEE Real-Time and Embedded Technology and Applications Symposium, IEEE (April 2007) 203–214
32. Mohaqeqi, M., Mousavi, M.R.: Sound test-suites for cyber-physical systems. In: 10th International Symposium on Theoretical Aspects of Software Engineering (TASE). (July 2016) 42–48
33. Kanade, A., Alur, R., Ivancic, F., Ramesh, S., Sankaranarayanan, S., Shashidhar, K.C.: Generating and analyzing symbolic traces of Simulink/Stateflow models. In: Proc. 21st International Conference on Computer Aided Verification (CAV), Springer (June 2009) 430–445

Real-Time Reachability for Verified Simplex Design

TAYLOR T. JOHNSON, University of Texas at Arlington
STANLEY BAK, Air Force Research Laboratory
MARCO CACCAMO, University of Illinois at Urbana-Champaign
LUI SHA, University of Illinois at Urbana-Champaign

The Simplex Architecture ensures the safe use of an unverifiable complex/smart controller by using it in conjunction with a verified safety controller and verified supervisory controller (switching logic). This architecture enables the safe use of smart, high-performance, untrusted, and complex control algorithms to enable autonomy without requiring the smart controllers to be formally verified or certified. Simplex incorporates a supervisory controller that will take over control from the unverified complex/smart controller if it misbehaves and use a safety controller. The supervisory controller should (1) guarantee the system never enters an unsafe state (safety), but should also (2) use the complex/smart controller as much as possible (minimize conservatism). The problem of precisely and correctly defining the switching logic of the supervisory controller has previously been considered either using a control-theoretic optimization approach, or through an offline hybrid systems reachability computation. In this work, we show that a combined online/offline approach that uses aspects of the two earlier methods along with a real-time reachability computation, also maintains safety, but with significantly less conservatism, allowing the complex controller to be used more frequently. We demonstrate the advantages of this unified approach on a saturated inverted pendulum system, where the verifiable region of attraction is over twice as large compared to the earlier approach. Additionally, to validate the claims that the real-time reachability approach may be implemented on embedded platforms, we have ported and conducted embedded hardware studies using both ARM processors and Atmel AVR microcontrollers. This is the first ever demonstration of a hybrid systems reachability computation in real-time on actual embedded platforms, and required addressing significant technical challenges.

Categories and Subject Descriptors: C.3 [**Computer systems organization**]: Embedded and cyber-physical systems

General Terms: Design, Verification

Additional Key Words and Phrases: formal verification,hybrid systems,cyber-physical systems

ACM Reference Format:
Taylor T. Johnson, Stanley Bak, Marco Caccamo, and Lui Sha, 2015. Real-Time Reachability for Verified Simplex Design. *ACM Trans. Embedd. Comput. Syst.* 1, 1, Article 1 (January 201X), 29 pages.
DOI : http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

Modern cyber-physical systems are large complex systems of systems, where arguments about the behavior of the whole system rely on guarantees about the individual components. Individual components, however, may be designed using machine learning methods such as neural networks that are currently not amenable to formal analysis, or the components may simply be too large and complex for complete verification. As such autonomy is incorporated into these increasingly smart systems that have the ability to learn from their environments and interactions through sophisticated complex/smart controllers, approaches are necessary to provide guarantees about their behavior.

One approach to provide formally verified behavior despite the use of unverified, complex, and smart control logic is the Simplex Architecture [Sha 2001]. Similar to how a driving instructor's car may have two steering wheels and two sets of brakes,

An earlier version of this work was presented at RTSS 2014 [Bak et al. 2014]. DISTRIBUTION A. Approved for public release; Distribution unlimited. (Approval AFRL PA case number 88ABW-2015-4618, 28 SEP 2015)

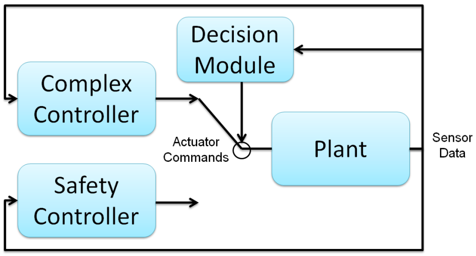


Fig. 1: The Simplex Architecture produces a verified system despite the use of an unverified complex/smart controller. The decision module should switch between the controllers to provide overall system safety.

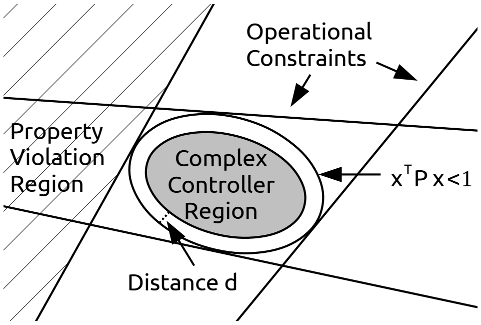


Fig. 2: The LMI Simplex design approach uses switching logic based on an ellipsoid within the system constraints in order to produce a verified system.

a Simplex system contains two controllers and supervisory switching logic. As long as the instructor intervenes to prevent dangerous situations, the untrusted student is allowed to drive. Similarly in Simplex, an unverified controller can actuate the system, as long as the verified one takes over quickly at potentially unsafe times.

In the Simplex Architecture, shown in Figure 1, unverified control logic (the complex/smart controller) is wrapped with a verified controller (the safety controller) and switching logic (the decision module). The complex/smart controller typically has better performance, or is concerned with mission critical requirements, whereas the safety controller is designed with simplicity and provability in mind, and may concern itself only with safety-critical aspects. When the system is in danger of entering an unrecoverable state, the decision module must switch control to the safety controller. In this way, the complex/smart controller can be used while still maintaining the formal guarantees of the safety controller. The key challenge when designing a system with the Simplex Architecture is to properly create the decision module logic.

It is easy to design safe decision module switching logic; one can simply always use the safety controller. This is undesirable, however, as mission-critical objectives might be delayed or ignored since the complex/smart controller is never used. The key challenge, which is the focus of this paper, is to *reduce the conservatism in the decision module design*. Control should not be switched too late, though, as the safety controller may not be able to safely recover the system.

In earlier Simplex designs, the switching logic was designed in one of two ways. From a control theoretic perspective, verified switching logic can be synthesized from the solution of a linear matrix inequality (LMI) along with the system dynamics and constraints [Seto and Sha 1999]. Alternatively, approaches based on hybrid systems reachability can be used to produce a provably safe decision module [Bak et al. 2011]. These earlier approaches will be reviewed in Section 2. In this paper, we propose the use of a unified approach, where the offline LMI result is combined with an online reachability computation to produce a *significantly less conservative* Simplex system that is still safe. We elaborate on this approach and prove its safety in Section 3.

The proposed approach requires computing reachability online for short time intervals. Previous hybrid systems reachability algorithms, however, were not designed for real-time computation and furthermore almost always require the use of numerous complex libraries for either performing simulations or for representing sets of reachable states as some geometric data structure (such as support functions, polytopes, zonotopes, symbolic expressions, etc.). For this reason, in Section 4, we propose a real-time reachability algorithm based on mixed face lifting [Dang 2000] that is compatible with the imprecise computation model in the real-time scheduling literature [Lin et al. 1987]. Real-time reachability has applications beyond Simplex, and is presented as a general online reachability approach. Next, we evaluate the proposed unified Simplex design in Section 5, on both x86 and embedded microprocessors. In order to provide a direct comparison, we use the existing system model from earlier Simplex work of an inverted pendulum system with saturation. The run-time approach significantly expands the space where the complex/smart controller may be used. Other research efforts related to Simplex and reachability are then presented in Section 6, followed by conclusions and directions for future work in Section 7.

2. BACKGROUND AND CONTRIBUTIONS

There have been several verified design methodologies for systems that use the Simplex Architecture. Before going into their details, we first present useful definitions.

The system is defined with a set of operational constraints, such as limits of actuators, physical restrictions, invariant safety properties that cannot be violated, or linearization boundaries where the model is considered valid.

DEFINITION 1. *States that do not violate any of the operational constraints are called* admissible states*. Those that violate the constraints are called* inadmissible states.

From this definition, we can define the set of states that are recoverable for a particular control strategy, assumed to be a given safety controller in the Simplex architecture.

DEFINITION 2. *The set of* recoverable states *is a subset of the admissible states, such that if the given safety controller is used from these states, all future states will remain admissible.*

The recoverable states are used in the switching rule instead of the admissible states due effectively to inertia in the system. That is, they are used to ensure that the safety controller and actuators have enough time to prevent the system from leaving the admissible states. Further, the intuition of defining the recoverable states as a subset of the admissible states is as follows. To enhance performance, we wish to stay within a small subset of highly desirable admissible states. The set of recoverable states is the subset of the set of admissible states that a safety control is guaranteed not to leave. However, the safety controller may not be able to keep the system inside the subset of recoverable states, namely the desirable states, and hence the complex/smart controller is needed. Their relation is illustrated in Figure 2, where the white ellipsoid is the recovery set.

With these definitions, we now describe two earlier approaches for verified Simplex design. The first is based on solving linear matrix inequalities (LMIs), and the second is based on reachability analysis of hybrid systems.

2.1. Verified Design using LMIs

The first proposed way to design a verified decision module is based on solving linear matrix inequalities (LMIs) [Seto and Sha 1999; Boyd et al. 1994], which has been used to design Simplex systems as complicated as automated landing maneuvers for an F-16 [Seto et al. 1999]. In this approach, system dynamics are approximated by a linear model using the standard control-theoretic approach, where $\dot{x} = Ax + Bu$ for state vector x and input u.

In this approach, the operational constraints, as well as saturation limits are expressed as linear constraints in an LMI. These constraints, along with linear dynamics for the system are input into a convex optimization problem that produces both linear proportional controller gains K as well as a positive-definite matrix P. The controller produced is a linear-state feedback controller, $u = Kx$, yielding the closed-loop dynamics $\dot{x} = (A + BK)x$. Given state x, when input Kx is used, the P matrix defines a Lyapunov potential function ($x^T Px$) which is positive-definite with negative-definite derivative (so it is monotonically decreasing over time), thus guaranteeing stability of the linear system using Lyapunov's direct or indirect (if the plant is nonlinear and was linearized) methods. Furthermore, the matrix P is constructed by the method such that it defines an ellipsoid in the state space where all the constraints are satisfied when $x^T Px < 1$. Since the states where saturation occurs were used as constraints in the method, any states inside the ellipsoid result in control commands that are not beyond the actuator limits (where saturation would occur).

In this way, when the gains K define the safety controller, the ellipsoid of states $x^T Px < 1$ is a subset of the recoverable states. The situation is shown visually in Figure 2. The feasible region is a subset of the admissible states defined by the input constraints (saturation), as well as the operational constraints. The stabilizable region (also known as the region of attraction) is the region of the state-space within which a given controller can stabilize the system. For the purpose of LMI-Simplex, this is also known as the recoverable region or the recoverable states as defined in Definition 2. For linear systems with constraints, this region may be under-approximated by solving an LMI of the determinant maximization form [Vandenberghe et al. 1998]. For a matrix that describes an ellipsoid $x^T Px = 1$, this has the effect of maximizing the product of the radii of the ellipsoid (which is related to the determinant of the matrix P). The volume of an ellipsoid, then, is proportional to this product. In this way, the optimization is maximizing the volume of the ellipsoid such that all states inside do not leave the ellipsoid, and all the constraints are satisfied for every state in the ellipsoid.

This approach is used to determine the proper behavior of the decision module. As long as the system remains inside the ellipsoid, any unverified, complex/smart controller can be used. If the state approaches the boundary of the ellipsoid, control can be switched to the safety controller that will drive the system towards the equilibrium point where $x^T Px = 0$. Care must be taken to ensure control is switched to the safety controller *before* the state leaves the ellipsoid. If the decision module simply checks the Lyapunov potential of the current state, then, once the state is outside of the ellipsoid, the system is not guaranteed to be recoverable without violating the operational constraints. Thus, a smaller subset of the state space must be used to define the states where the complex controller is allowed to actuate the system. In Figure 2, the distance d defines this extra buffer that can be determined offline by computing the maximum gradient for any control command inside the ellipsoid, multiplied by the period of the decision logic. As long as d is no smaller than the maximum distance traveled in the state-space over the time of one full control period, then d is large enough to ensure switching to the safety controller can recover the system.

For safety it is sufficient to consider only a single switch to the safety controller and never switching back. If switching back is desired, this should not be done arbitrarily as the composed switched system might be unstable. Specifically, the safety controller should be used at least until a state within the complex/smart controller region (as shown in Figure 2) is reentered, before switching back to the complex/smart controller.

2.2. Verified Design using Reachability

An alternative method for verified Simplex design is based on reachability analysis of hybrid systems [Bak 2013b], which has been used, for example, to create a Simplex system to prevent off-road vehicle rollover [Bak et al. 2010]. In this approach, the dynamics are defined using a hybrid automaton, which is a formal model for a system with both continuous and discrete behaviors. Mathematically, a hybrid automaton [Alur et al. 1995] is a tuple, $H = (\mathcal{X}, L, X_0, I, F, T)$, where:

- $\mathcal{X}$ is the set of continuous states. For a system with n real-valued dimensions, the continuous state is $\mathbb{R}^n$.
- L is the set of discrete states (locations). The state of a hybrid automaton is an element of $X = L \times \mathcal{X}$.
- X_0 is a set of initial states, which is a subset of X.
- I is a set of invariants that defines the continuous states that are possible for each location. It is a function $L \rightarrow 2^{\mathcal{X}}$.
- F is a set of flows, each of which defines the differential equations in each location. It is a function $X \rightarrow 2^{\mathbb{R}^n}$.
- T is a set of discrete transitions, each of which defines switching between discrete locations. A transition is composed of a guard condition for when the transition is enabled, and a reset map that can reassign the continuous states from the predecessor mode to the successor mode. In general, it is a relation $T \subseteq X \times X$.

Semantically, a hybrid automaton behaves by advancing time according to the differential equations defined in the mode of the current discrete state $l \in L$, then allowing any enabled transitions to be taken, and repeating, yielding a sequence of states called an *execution*. A state $x \in X$ is *reachable* is there exists a finite execution ending in x. The *set of reachable states* contains every reachable state. The guard conditions on the outgoing transitions define when the location can change. The invariants of the locations can be used to force transitions by preventing time from elapsing further in the current mode. Together, these allow nondeterminsim in the discrete behavior. A hybrid automaton can be graphically depicted as a finite-state machine with differential equations in each discrete state. The model also allows for nondeterminism in the continuous behavior because a single state $x \in X$ may be associated a set of derivatives for each variable, via the set of flows F.

This modeling framework is very expressive, and computing exactly the sets of states a hybrid automaton may enter, called the *reachable* set of states, is undecidable [Henzinger et al. 1995]. Thus, analysis of hybrid systems often restricts either the continuous dynamics or the discrete dynamics [Alur and Dill 1994; Lafferriere et al. 2000; Branicky 1998]. In this paper, the reachability algorithm proposed in Section 4 considers restricted hybrid automata models where (a) the state invariants are disjoint and cover the continuous states $\mathbb{R}^n$, (b) there are no reset maps in the transitions between discrete states, and (c) the guards of incoming transitions are defined by the state invariants.

In addition to restrictions on dynamics, practical reachability approaches often over-approximate the set of reachable states [Kapinski and Krogh 2002; Dang et al. 2010; Frehse et al. 2011], which is sufficient for proving safety properties. If a sound over-approximation of the reachable set of states for a hybrid automaton does not contain any unsafe states, then the system is verified as safe since no unsafe states are in the actual reachable set of states either. That is, the system is safe if the intersection of the over-approximation of the set of reachable states and the set of unsafe states is empty. This approach may, however, lead to spurious counterexamples where the error due to the over-approximation contains unsafe states, but the actual reachable set of states does not.

We define $\textsc{Reach}_{\infty}(x, \textsc{ha})$ to be the set of states reached *in any amount of time* from state x in hybrid automaton $\textsc{ha}$, $\textsc{Reach}_{\leq t}(x, \textsc{ha})$ is the set of states reached from x in *up to* t time, and $\textsc{Reach}_{=t}(x, \textsc{ha})$ is the set of states reached after *exactly* t time has elapsed. Also, we naturally extend $\textsc{Reach}$ to initial *sets* of states, where the resultant set of reachable states is the union of the set of reachable states from each state in the initial set.

In terms of Simplex design, the behavior of an optimal decision module can be defined in terms of reachability. Optimal here means that the given safety controller takes over only if it has to; if it did not take over, then the system remains in admissible states and can enter the subset of recoverable states that can be pre-computed offline e.g., using LMIs. Furthermore, it never takes over when the complex/smart controller could safely be used. The switching condition (formalized as the transition's guard and invariant in the hybrid automaton) between the safety controller and complex/smart controller modes is defined using the following theorem [Bak et al. 2011].

THEOREM 3. *The optimal switching condition for Simplex is given when, at every control iteration, the complex/smart controller is used if and only if (1)* $\textsc{Reach}_{\leq\delta}(x, \textsc{cc}) \cap U = \emptyset$ *and (2)* $\textsc{Reach}_{\infty}(\textsc{Reach}_{=\delta}(x, \textsc{cc}), \textsc{sc}) \cap U = \emptyset$*, where* $x \in X$ *is the current state and* $U \subseteq X$ *is the set of inadmissible (unsafe) states.*

The inner $\textsc{Reach}_{=\delta}$ in part (2) is the time-bounded reachability of the system for one decision logic switching interval time, δ, while using the complex/smart controller ($\textsc{cc}$). The outer $\textsc{Reach}_{\infty}$ is the infinite-time reachability for the system under control of the safety controller ($\textsc{sc}$).

Intuitively, this check is examining what happens if the complex/smart controller is used for a single control interval of time δ, and then the safety controller is used thereafter. If this set of states contains an inadmissible state (either before the switch as in part (1) or after as in part (2)), then the complex/smart controller cannot be used for one more control interval, and instead the safety controller must be used right away. Assuming the system starts in a recoverable state, this guarantees it will remain in the recoverable set for all time.

Several factors prevent the direct use of Theorem 3. The first is that the reason to apply Simplex is that a precise model of the complex/smart controller is not available, but rather an over-approximation must be used which can be computed, for example, based on the plant model and actuator limits. Second, as discussed before, computing reachability exactly for a general hybrid automaton is undecidable. However, estimates of the set of recoverable states (Definition 2), can still be computed using over-approximations, where the conservativeness of the resultant decision module depends on the amount of over-approximation. Third, the switching condition is defined in terms of a specific state x, which is not useful for offline computation since every state would need to be enumerated. Instead, the condition can be rewritten in terms of backwards reachability from the set of inadmissible states, which can then be computed offline [Bak et al. 2011; Bak 2013b]. As with the LMI approach, the output is a set of states which forms a guaranteed subset of the recoverable states.[1]. These considerations are combined in order to provide a condition for effectively computing the decision module logic as follows.

COROLLARY 4. *A safe switching condition for Simplex is given when, at every control iteration, the complex/smart controller is used if the current state* $x \notin \textsc{BackReach}^{*}_{\leq\delta}(\textsc{BackReach}^{*}(U, \textsc{sc}), \textsc{cc}')$.

Here, $\textsc{BackReach}^{*}$ is an over-approximation of the exact set of backward reachable states for all time (that is, to a fixed-point). The inner $\textsc{BackReach}^{*}$ defines the states

[1] The set of backward reachable states can be computed for deterministic systems (including linear systems) by negating the differential equations and inverting the transitions in the hybrid automaton and using standard forward reachability techniques. This technique is known as back-reachability.

where, if the system were to start from the set of unsafe states U and use the safety controller, it could still violate one or more of the safety constraints. Then, the outer $\textsc{BackReach}^*_{\leq\delta}$ is the set of states that, within one control interval, can reach an unrecoverable state. Since the unrecoverable states contain the unsafe states, and since the outer $\textsc{BackReach}^*_{\leq\delta}$ checks up to δ time rather than exactly δ time, a separate condition is not needed to check if the the complex/smart controller itself reaches the unsafe states, as in part (1) of Theorem 3.

The pessimism in the resultant decision logic depends on both the accuracy of the reachability computation as well as on how much $\textsc{cc}'$ over-approximates the exact complex/smart controller model $\textsc{cc}$. The condition in Corollary 4 is more useful than the one in Theorem 3 because it can be effectively computed using existing hybrid systems reachability algorithms. The set of states on the right-hand side can be computed offline and encoded in some form (for example, using linear bounds [Bak 2009]) and then, online, the decision module need only check if the current state exists within the encoded set of states. If it does, then the safety controller must be immediately used. If it does not, then the complex/smart controller can be used for one control interval, after which the condition will be checked again on the new state.

2.3. Contributions

In this paper, building on our prior work [Bak et al. 2014], we show how to combine the LMI-based Simplex method with a real-time reachability method into a unified framework to ensure safety while drastically decreasing the overconservative use of the safety controller. Specifically, if it is possible to use the set of recoverable states computed using the LMI method for the switching condition, we do so. If not and the system is at a state outside the recoverable states based on the LMI ellipsoids, then we try to check safety using a novel real-time reachability method, in contrast to the previous offline reachability approach. Together, we illustrate how this unified approach gives both real-time guarantees and reduces conservatism of when the safety controller is used. A main contribution of our approach is the first ever demonstration of a reachability method in real-time, enabled by our careful design and implementation that does not use any dynamic memory allocation nor rely on sophisticated (non-portable) libraries that many other methods use, such as the Parma Polyhedral Library (PPL) [Bagnara et al. 2008], recent satisfiability-modulo theories (SMT) approaches [Gao et al. 2013], or validated integration tools [Duggirala et al. 2013]. To validate the feasibility of actually implementing the method in real-time embedded hardware, we have ported our prototype method from [Bak et al. 2014] that was implemented on x86-64 platforms to several embedded platforms (namely a 32-bit ARM-based system and an 8-bit Atmel AVR ATmega32u4-based Arduino system). This effort validates our claims from [Bak et al. 2014], which were not previously validated in embedded hardware. *The key result of this paper is the first ever demonstration of a hybrid systems reachability algorithm implemented in embedded hardware that can meet real-time guarantees, which required carefully designing the reachability algorithm as described in this paper.* We have additionally added significant further details of the approach and case study to the paper over [Bak et al. 2014], including code snippet examples for the case study.

3. UNIFIED APPROACH FOR SIMPLEX DESIGN

The two existing approaches for Simplex design previously discussed each have their own limitations. The LMI approach works when the system model is linear. If there are actuator limits, and the input to the actuators u (from $\dot{x} = Ax + Bu$) can saturate, the output of the optimization will be a set of states where the command used by the safety controller is within the saturation limits. This is done by adding a constraint based on the state-feedback gain as part of the optimization (the input is $u = Kx$, which is bounded by the linear constraints $Kx \leq$ MAX_INPUT and $Kx \geq$ MIN_INPUT).

The set of states output by the LMI approach is safe, but may be pessimistic, since a saturated safety controller may still be able to recover the system. Furthermore, the resultant switching condition is based on a Lyapunov function which—due to convexity and quadratic restrictions required in the optimization algorithms—has level sets that are ellipsoidal. This is a sufficient but not necessary condition for stability and therefore the switching set is almost certainly conservative. We demonstrate this pessimism in our evaluation in Section 5.

The reachability-based Simplex approach is not restricted to linear systems, and can have its conservatism decreased by increasing the accuracy of the reachability computation[2]. One downside of this approach is that over-approximation error occurs from the need to abstract the complex/smart controller hybrid automaton by a hybrid automaton which takes into account any possible complex/smart controller command. A second issue is the difficulty of succinctly and accurately encoding the result of the computation, which in general may be a large non-convex set in many dimensions. Lastly, hybrid systems reachability methods introduce over-approximation error, which can be large when the initial set of states is large and the reachability time bound is large. The back-reachability formulation of Theorem 3 includes a time-unbounded reachability computation from the set of inadmissible states, which can require significant computation time.

We now present an alternative design for a verified Simplex system. The proposed technique makes use of aspects from both of the previous verified design approaches in order to overcome some of their individual limitations.

First, we formalize the connection of the ellipsoid from of the LMI approach with that of a reachability computation of a hybrid automaton (which by the ellipsoid's construction remains in a single, unsaturated mode):

LEMMA 5. *The output of the LMI approach, the potential function P and controller gains K, define a safety controller* SC *and a subset of the recoverable set of states $\mathcal{R} = \{x | x^T P x < 1\}$, where* $\text{REACH}_{\infty}(\mathcal{R}, \text{SC}) \cap U = \emptyset$.

This is true because the potential function is guaranteed to satisfy the constraints passed to the LMI solver, including avoidance of the inadmissible states, when $X^T P X < 1$. When the controller gain vector K output by the approach is used (which defines the safety controller update $u = Kx$), the potential function is strictly decreasing over time (i.e., it is a Lyapunov function). Therefore, it is guaranteed for unbounded time that any state starting inside $\mathcal{R}$ will remain inside $\mathcal{R}$. Since there are no inadmissible states in $\mathcal{R}$, no inadmissible states will ever be reached.

We can now define an alternate condition for safe switching logic:

THEOREM 6. *A safe switching condition for Simplex is given when, at every control iteration, the complex/smart controller is used if, for some α time, (1)* $\text{REACH}_{\leq\delta}(x, \text{CC}) \cap U = \emptyset$, *(2)* $\text{REACH}_{\leq\alpha}(\text{REACH}_{=\delta}(x, \text{CC}), \text{SC}) \cap U = \emptyset$, *and (3)* $\text{REACH}_{=\alpha}(\text{REACH}_{=\delta}(x, \text{CC}), \text{SC}) \subseteq \mathcal{R}$.

PROOF. Intuitively, this switching condition states that the complex/smart controller may be used if: (1) the complex/smart controller cannot reach an unsafe state before the next decision interval (at time δ), (2) if the safety controller takes over at the next decision interval, it will avoid unsafe states until $\delta + \alpha$ times passes, and (3) after $\delta + \alpha$ time, a state in $\mathcal{R}$ will be reached.

[2] Over-approximating reachability approaches typically have an accuracy / computation time trade off.

More formally, assume by contradiction that this is not a safe switching condition, so an inadmissible state is reached at some time. This time will be either less than δ, more than δ and less than $\delta + \alpha$, or more than $\delta + \alpha$. The first two of these cases are ruled out directly by conditions (1) and (2), so only the third case needs to be examined.

From Lemma 5 $\textsc{Reach}_\infty(\mathcal{R}, \textsc{sc}) \cap U = \emptyset$. If $\mathcal{R}' \subseteq \mathcal{R}$, $\textsc{Reach}_\infty(\mathcal{R}', \textsc{sc}) \subseteq \textsc{Reach}_\infty(\mathcal{R}, \textsc{sc})$, then the smaller set of states $\mathcal{R}' = \textsc{Reach}_{=\alpha}(\textsc{Reach}_{=\delta}(x, \textsc{cc}), \textsc{sc}) \subseteq \mathcal{R}$ will also satisfy the condition $\textsc{Reach}(\mathcal{R}', \textsc{sc}) \cap U = \emptyset$. Therefore, every state reached after $\delta + \alpha$ is also admissible.

Since all three cases do not contain an inadmissible state, our assumption that an inadmissible state is reached is violated, yielding a contradiction, and therefore this is a safe switching condition. □

In summary, the proposed approach is as follows: when the system is well-inside the ellipsoid that represents the largest safe sublevel set of the Lyapunov function, we do not need to invoke an extensive reachability analysis using the safety controller, as we know the state is recoverable (even for the next control period). When the system state is near the boundary of the ellipsoid, the reachability analysis is used to allow the system to cross the boundary of the ellipsoid as long as the reachability computation shows that (1) no system constraints are violated when this is done (i.e., none of the reachable states violate a system constraint), and (2) the state can be guaranteed to be brought back into the ellipsoid (i.e., the reachable states return inside the ellipsoid). This allows the complex controller to be used in a larger region compared with the LMI-approach because it can soundly reason about the behavior of the system outside of the ellipsoid (remember that the Lyapunov function from the LMI method is only a sufficient condition for safe switching). This condition can also be less conservative than the pure reachability approach because the computation needed is from a single state x, rather than the possibly large set of inadmissible states. Additionally, it involves reasoning over a finite-time horizon ($\alpha + \delta$), rather than infinite-time reachability needed in the method based on Theorem 3.

There are still two issues which need to be addressed before the condition in Theorem 6 is usable. First, since we cannot compute reachability exactly for complex hybrid automata due to decidability reasons [Henzinger et al. 1995], we will instead compute an over-approximation. This will result in a conservative switching set depending on the accuracy of the computation. Second, this computation is defined from the system's current state x, which is not available offline. In order to resolve this issue, we propose an online, real-time reachability computation method in the next section. After that, in Section 5, we will evaluate the conservatism in the switching set due to the over-approximation in the proposed algorithm.

4. REAL-TIME REACHABILITY ALGORITHM

Hybrid systems reachability computations have been traditionally computed offline, and are both memory and processor intensive operations. In Section 3, we have illustrated several reasons to perform the reachability computation at runtime. This requires a reachability algorithm capable of use within a real-time system. In this section, we describe a real-time reachability algorithm with the following key features:

- High-performance for a quick runtime for short reachability times.
- The ability to check the three conditions from Theorem 6.
- No dynamic data structures (or large memory preallocation) or recursion, for usability in a real-time system.
- No dependence on complex external libraries (only the C standard library) that most if not all other reachability approaches use.
- Iterative improvement in accuracy with increased computation time.

The last point is important because it allows the reachability task to be scheduled in the framework of imprecise real-time system computation [Liu et al. 1994]. In this framework, each task produces a partial result that is usable and improved upon as more computation time is added (this is sometimes called an *anytime algorithm*). In particular, the proposed reachability algorithm is based on the milestone approach [Lin et al. 1987], where partial results are recorded at various points during the execution, and the last-recorded values are used when the final result is needed. This is in contrast to the traditional real-time systems execution model where each task has a fixed worst-case execution time (WCET) [Liu and Layland 1973].

We now present the real-time reachability algorithm that is suitable for real-time, online, computation that satisfies the above requirements. We distinguish between *reach-time*, which is the time we are computing reachability for, and *runtime*, which is the duration of (wall) time the method is allowed to run. Recall that the types of hybrid systems we consider are ones where the state invariants are disjoint and cover the continuous state $\mathbb{R}^n$, there are no reset maps in the transitions between discrete states, and the guards of incoming transitions are defined by the state invariants. In these piecewise systems, the state of the hybrid automaton can be determined solely by the continuous state, although different differential equations can be used in different parts of the state space. This is applicable to many state-feedback continuous systems with saturation (such as those using gain scheduling controllers) since the states where saturation occurs are typically disjoint from the unsaturated states (because the actuator command is a function of the state), and the continuous states do not jump along the saturation boundary.

To employ the real-time reachability algorithm, as in our earlier work [Bak et al. 2011], the user defines the system dynamics through a function (a function written in the C language in this implementation) that returns the minimum and maximum derivative in each dimension given an arbitrary box of the state space. The derivative needed in the algorithm is always in the outward direction of the box of states being tracked. The tracked box has $2n$ faces, where n is the number of dimensions. For each of the n dimensions, these faces are represented by a minimum face, and a maximum face. That is, there are total $2n$ minimum and maximum faces, each of which refers to particular faces of a hyperrectangle used to represent portions of the set of reachable states. If the minimum face is being considered, the minimum of the derivative is used, as this may (but not necessarily so) push the tracked states outward from the hyperrectangle. If the maximum face is being considered, the maximum of the derivative is used for the same reason. Nonlinear dynamics are permitted in this approach, so long as the user-provided function maximizes or minimizes the nonlinear derivatives within an arbitrary box. Notice that this does not require solving the differential equations (which is generally a harder problem), since the bounds are on the derivatives themselves. Furthermore, we require the derivatives are defined in the entire state space, and that they are bounded.

The real-time reachability algorithm is based on mixed face lifting [Dang and Maler 1998; Dang 2000]. This approach is a *flow-pipe construction* method, which means that snapshots of the reachable set of states are computed at increasing points in reach-time, and reasoning is done about which states can be encountered between snapshots.

To create a real-time implementation, we use boxes (n-dimensional hyperrectangles) as our representation of the set of states. Over long reach-times, this representation can be problematic because, if the actual reachable set of states is not a box, error is introduced by over-approximating it as one (called the wrapping effect [Moore 1966]). However, since we only need to compute reachability for short reach-times ($\delta+\alpha$ from Theorem 6), a simpler, faster, representation is preferred to better long-term error control. In mixed face lifting, the dynamics along each face are over-approximated by

Algorithm 1 The real-time reachability algorithm uses a desired reach-time step to tune its runtime based on the available computation time.

```
Box currentBox := initialBox

while (reachTimeRemaining > 0)
  Box[] nebs = constructNeighborhoods(currentBox, reachTimeStep)

  crossReachTime := minCrossReachTime(nebs)
  advanceReachTime := min(crossReachTime, reachTimeRemaining)
  currentBox := advanceBox(nebs, advanceReachTime)

  reachTimeRemaining := reachTimeRemaining − reachTimeToAdvance
end while
```

the maximum derivative along that face. The reach-time is then advanced uniformly along all faces (i.e., in all directions).

We modify the original mixed face lifting algorithm to make it usable in a real-time setting. In particular, instead of using the desired error in order to control the neighborhood width around each face [Dang 2000], we use a desired reach-time step to control neighborhood widths. This parameter allows us to tune the total number of steps used in the method, and therefore alter the runtime. After the given reach-time is obtained, the desired step size is decreased (which reduces the width of the neighborhoods, and therefore the derivative error at each step) and the computation is restarted. In our algorithm, initially we use a time step which is some factor, say one tenth, of the desired reach time. The decrease is computed by dividing the time step by two. In this way, the algorithm will produce progressively more accurate answers, for as much runtime as the task is given.

The high-level algorithm, given a fixed desired step size (`reachTimeStep`), is given in Algorithm 1. For a box, there are two faces for every dimension (one for each of the minimum and maximum faces along that dimension), and there are two corresponding face neighborhoods (regions where the face may advance through during the current time step) for every dimension. The neighborhoods, `nebs`, are constructed based on the desired reach-time step. This neighborhood construction process will be elaborated on shortly.

Next, the minimum reach-time for any point along each face to cross the corresponding neighborhood in the corresponding direction is computed. What this means is that, for example in the two-dimensional example of Figure 3, the minimum reach-time for any point along the left face of `currentBox` to cross to the left side of `nebs[0]` in the x direction is computed, as well as the minimum reach-time for any point along the right face to cross `nebs[1]`, as well as the neighborhoods in the y directions, and then the minimum of all of these is returned. This is computed by looking at the minimum or maximum derivative within the box for each neighborhood (from the user-provided derivative bounds function), as well as the width of the neighborhood along the corresponding dimension.

Finally, the `currentBox` at the next reach-time step is computed based on the neighborhoods and computed reach-time to advance (which may be reduced if it exceeds `reachTimeRemaining`). This is done by advancing each face by the maximum derivative in the outward direction in its neighborhood (from the user-provided derivative bounds function) multiplied by `advanceReachTime`.

The novel aspect of this face lifting reachability algorithm is that the widths of the neighborhoods are tunable by the `reachTimeStep` parameter. The neighborhood construction (the `constructNeighborhoods` function) proceeds in three steps:

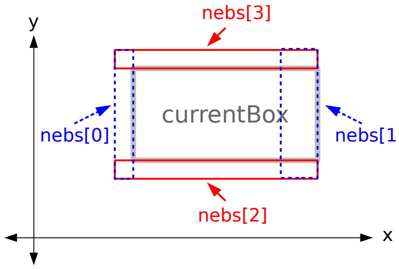


Fig. 3: The neighborhood widths are determined by `reachTimeStep` and the derivatives along the faces of `currentBox`.

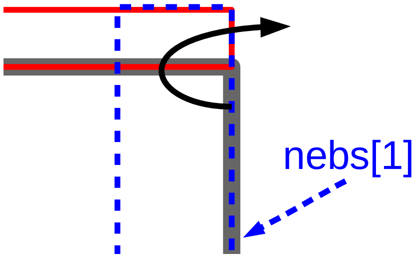


Fig. 4: Although the derivative along the face may be inward-facing, the derivative in the neighborhood can still be outward facing. The first condition of step 3 in the neighborhood construction process checks for this and reconstructs the neighborhoods if such a situation occurs. Here, `nebs[1]` would be updated to an outward-facing neighborhood, which would require subsequent reconstruction of the other neighborhoods (because the edges overlap).

(1) The maximum outward derivative along each face of `currentBox` is computed. One neighborhood is constructed for each face, where the width of the corresponding neighborhood is based on the derivative (the width is the derivative multiplied by the passed-in desired `reachTimeStep`).
(2) The neighborhood boxes are all constructed based on the computed widths, such that the edges overlap as shown in Figure 3. We call a neighborhood constructed on the inside of the corresponding face an *inward-facing* neighborhood (such as `nebs[1]` in the figure).
(3) The outward derivatives in the constructed neighborhoods are computed with the user-provided function. If either (1) an inward-facing neighborhood contains an outward-facing derivative, or (2) a derivative has doubled in value since the previous derivative computation for that neighborhood (which initially is the flat neighborhood), the width of the neighborhood is recomputed and the process repeats by returning to step 2.

The check in step 3 ensures two things. The first condition is necessary in case a derivative was inward-facing in a previously-constructed neighborhood, but outward-facing in the new, larger neighborhood. This case is shown visually in Figure 4. The second condition guarantees that the reach-time to progress from a point on the face through the corresponding face neighborhood is at least `reachTimeStep`/2. Due to this, we can bound the maximum number of iterations of the while loop as the desired reach time divided by `reachTimeStep`/2. Since the edges of the neighborhoods overlap, the neighborhoods of the other faces need to be reconstructed as well, which is why the algorithm backtracks to step 2.

The number of times the neighborhood construction backtracks from step 3 to step 2 is also bounded. This is because a face can flip from inward-facing to outward-facing only once, and since it was assumed there is a maximum derivative in the state space, the observed derivative can only double a finite number of times.

The imprecise computation version of the algorithm proceeds by running Algorithm 1 repeatedly, decreasing `reachTimeStep` after each repetition. In our implementation, after each execution `reachTimeStep` was halved, although strategies other than halving are also possible (this is a trade off between the time between milestones and the error reduction obtained at each iteration). When the deadline is reached (or the real-time reachability task is stopped), the most recent reachability result is used as the output. For this reason, the exact number of iterations of the neighborhood construction loop is not too useful, as long as it has an upper bound, and we can adjust it with `reachTimeStep`.

If the derivative doubles several times, the tracked box will be pessimistic, since the conservatism comes from over-approximating a derivative in a neighborhood by its maximum value. For this reason, we also set a threshold in the loop for how large the tracked boxes are allowed to get (not shown), and if it is exceeded we immediately halve `reachTimeStep` and restart the loop. If the number of backtracks to step 2 is small (which is true in practice), each advancement of time takes $O(n)$ where n is the number of dimensions in the system.

From the four desired properties of a real-time reachability algorithm mentioned earlier, this algorithm is quick (no exponential complexity operations), requires no dynamic memory or recursion, and can iteratively provide a better answer. In order to satisfy the remaining desired condition, we need to provide the ability to check the three conditions from Theorem 6. Rather than first computing the reachable set of states and then checking the conditions in that set (which would require dynamic storage to store the reachable set), we instead modify the core algorithm in Algorithm 1 to do the checks during the computation. Conditions (1) and (2) of the theorem deal with the safety of reachable states at intermediate reach-times. This can be checked inside the while loop by taking the convex hull of `currentBox` before and after the `advanceTime` call, and passing that to a function which ensures the hull does not contain a state which violates the system constraints. For checking condition (3), the final `currentBox` value can be used. Furthermore, these checks can be done at each iteration of the refinement; if a `reachTimeStep` is found such that the three conditions of the theorem are satisfied, no further refinement is necessary (and the complex/smart controller can be used).

5. EVALUATION

We now present an evaluation of the proposed methodology.[3] The real-time reachability approach computes the set of reachable states for the safety controller as depicted in the automaton representing the Simplex architecture in Figure 5. We demonstrate the method through two related case studies: a nonlinear inverted pendulum and a linear inverted pendulum. As another benefit of the real-time reachability method described in this paper, it can also work even if the LMI approach cannot be used. We note that the LMI approach in general cannot be used for nonlinear systems, so its application is limited. In order to directly show the advantage of the approach in the linear case, we use the same case study that demonstrated the earlier, LMI-based Simplex work [Seto and Sha 1999]. The linear inverted pendulum model is obtained by linearizing the nonlinear inverted pendulum model, and overall, their results are comparable and were used to validate against one another. Both models are briefly discussed here, with more details on the nonlinear and linear models in the earlier report [Seto and Sha 1999]. The system is an inverted pendulum with state constraints and input saturation. The physical system is shown in Figure 6 and consists of a DC-motor driven cart that moves along a 1-d track with a pendulum arm attached by an

[3] As it is difficult to present all the details necessary to replicate our results in the form of a paper, the source code implementation of the real-time reachability algorithm and our models are available as supplementary material and online at: http://www.verivital.com/rtreach/.

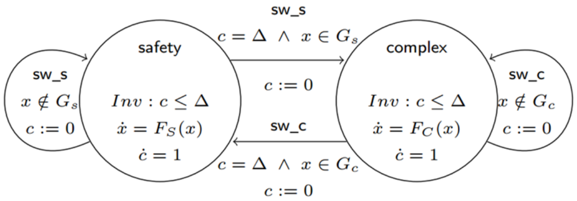


Fig. 5: Hybrid automaton for the Simplex architecture on which the real-time reachability computation is performed for the safety controller mode, where a formal model is available. Here Δ is a control period, c is a timer, G_S is a guard governing the transition from the safety to complex controllers, G_c is a guard governing the transition from the complex to safety controllers, and F_S and F_C respectively denote the dynamics of the overall closed-loop system when using the safety and complex controllers.

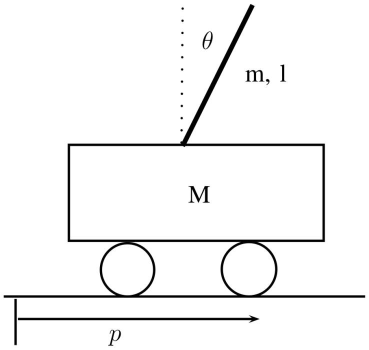


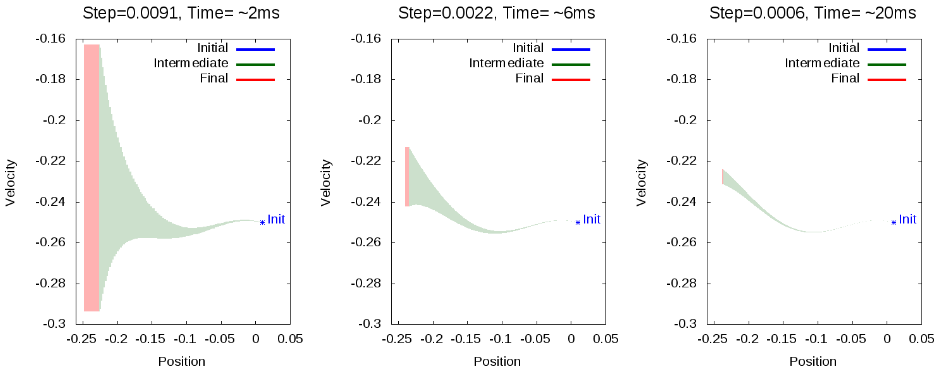
Fig. 6: An inverted pendulum system keeps a rod upright at an unstable equilibrium point by controlling a cart at its base.

angular joint to the cart. The control objective is to keep the angle θ of the pendulum arm at 0° measured from the vertical (i.e., to keep the arm upright).

There are four state variables: cart position p, cart velocity $v = \dot{p}$, pendulum arm angle θ, and pendulum arm angular velocity $\omega = \dot{\theta}$. We denote x as the state vector and p as the position, seen together next in Equation 1:

$$x = \begin{bmatrix} p \\ v \\ \theta \\ \omega \end{bmatrix} = \begin{bmatrix} p \\ \dot{p} \\ \theta \\ \dot{\theta} \end{bmatrix}, \text{ yielding the dynamics } \dot{x} = \begin{bmatrix} \dot{p} \\ \ddot{p} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} v \\ \dot{v} \\ \dot{\theta} \\ \dot{\omega} \end{bmatrix}. \quad (1)$$

The system is subject to physical constraints. The range of p is between $[-1, 1]$ meters, $\dot{p}$ is between$[-1.0, 1.0]$ meters/second, θ is between $[-15, 15]^\circ$, and $\dot{\theta}$ is unconstrained although the constraints on $\dot{p}$ do impose limits on $\dot{\theta}$). We ignore static friction (with respect to the cart wheels and ground, and with respect to the pendulum arm and joint) and take the armature inductance ($L_a = 18$ millihenries) to be 0 henries hence reducing the order of the system by making the armature current state variable I_a a function of only V_a. Without this simplification, two control states would be necessary.



(a) (b) (c)

Fig. 7: Overapproximation of the set of reachable states computed by the real-time reachability method for the nonlinear inverted pendulum model Equation 2 and Equation 3 for different amounts of computation runtime, 2 ms in (a), 6 ms in (b), and 20 ms in (c).

5.1. Nonlinear Inverted Pendulum

The inverted pendulum's state evolves according to a nonlinear differential equation $\dot{x} = f(x, u)$. Specifically,

$$f(x,u) \; = \begin{bmatrix} \dot{p} \\ \ddot{p} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} v \\ -\frac{\frac{1}{3} l^2\, m\,(C_1+f_c) - \frac{1}{2} l\, m\, \cos\,(\theta)\,(C_2+f_p)}{D} \\ \omega \\ -\frac{\bar{M}\,(C_2+f_p) - \frac{1}{2} l\, m\, \cos\,(\theta)\,(C_1+f_c)}{D} \end{bmatrix}. \tag{2}$$

Here, $D_l = 4\bar{M} - 3m$, $\bar{B} = \frac{K_g B_m}{r^2} + \frac{K_g^2 K_i K_b}{r^2 R_a}$, $B_l = \frac{K_g K_i}{r R_a}$, $\bar{M} = \frac{m+M+(K_g J_m)}{r^2}$, $f_c = B_x\, v + A_x\, e^{-C_x\,|v|}\, \text{sign}\;(v)$, $f_p = B_\theta\, \omega + A_\theta\, e^{-C_\theta\,|\omega|}\, \text{sign}\;(\omega)$, $D = l^2\, m\left(\frac{M}{3} + \frac{m}{3} + \frac{J_m\, K_g}{3\, r^2}\right) - \frac{l^2\, m^2\, \cos\,(\theta)^2}{4}$, $C_1 = v\left(\frac{B_m\, K_g}{r^2} + \frac{K_b\, {K_g}^2\, K_i}{R_a\, r^2}\right) - \frac{l\, m\, \omega^2\, \sin\,(\theta)}{2}$, and $C_2 = -\frac{g\, l\, m\, \sin\,(\theta)}{2}$. The pendulum model involves the following parameters: g is gravity, R_a is the armature resistance, r is the driving wheel radius, J_m is the motor rotor inertia, B_m is the motor's coefficient of viscous friction, B_θ is the pendulum joint's coefficient of viscous friction, K_i is the motor torque constant, K_b is the motor back-e.m.f. constant, K_g is the gear ratio, M is the cart mass, m is the pendulum arm mass, l is the pendulum arm length, f_c is the static friction force, and f_p is the viscous friction force. After evaluating values for constant parameters (the same as those used in [Seto and Sha 1999]), we have:

$$f(x,u) \; = \begin{bmatrix} \dot{p} \\ \ddot{p} \\ \dot{\theta} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} v \\ -\frac{0.020833\omega^2 \sin(\theta) - 0.059221v + 0.25\cos(\theta)(0.0001\omega + 2.45\sin(\theta))}{0.0625\cos(\theta)^2 - 0.604167} \\ \omega \\ \frac{0.000725\omega + 17.7625\sin(\theta) - 0.25\cos(\theta)(-0.25\sin(\theta)\omega^2 + 0.710657v)}{0.0625\cos(\theta)^2 - 0.604167} \end{bmatrix}. \tag{3}$$

As illustrated in Figure 7 by the decreasing size of the overapproximation of the set of reachable states, the more runtime given to the real-time reachability algorithm, the more accurate the result (see also Figure 8 and Tables I, II, and III). These results illustrate that the real-time reachability algorithm presented in this paper is effective even for hybrid systems with nonlinear differential equations. Thus, the results are widely applicable to many realistic systems.

5.2. Linearized Inverted Pendulum

As discussed in Section 5.1, the system is in general nonlinear, $\dot{x} = f(x, u)$, but to apply the LMI-Simplex approach as a part of the unified Simplex method described in this paper, we next work with a model linearized around the origin, which is the equilibrium point:

$$\dot{x} = Ax + Bu.$$

The linearization is justified since the control objective is to stabilize the system in a neighborhood of the vertical equilibrium, defined in this coordinate system as $\theta = 0°$, which is at the origin.

The plant system matrix and input vector used in Equation 5.2 are:

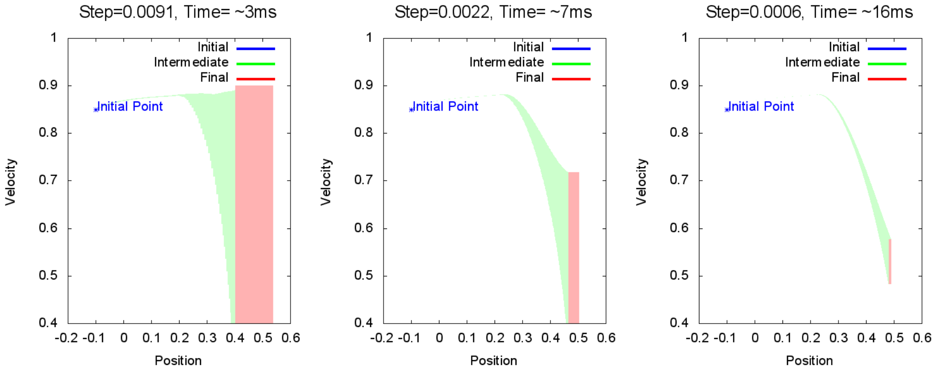
$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -a_{22} & -a_{23} & a_{24} \\ 0 & 0 & 0 & 1 \\ 0 & a_{42} & a_{43} & -a_{44} \end{bmatrix} \text{ and } B = \begin{bmatrix} 0 \\ b_2 \\ 0 \\ -b_4 \end{bmatrix}, \tag{4}$$

where $a_{22} = \frac{4\bar{B}}{D_l}$, $a_{23} = \frac{3mg}{D_l}$, $a_{24} = \frac{6B_\theta}{lD_l}$, $a_{42} = \frac{6\bar{B}}{lD_l}$, $a_{43} = \frac{6\bar{M}g}{lD_l}$, $a_{44} = \frac{12\bar{M}B_\theta}{ml^2D_l}$, $b_2 = \frac{4B_l}{D_l}$, and $b_4 = \frac{6B_1}{lD_l}$, for all the parameters defined in Section 5.1. Using the parameters from the earlier Simplex report [Seto and Sha 1999], the A and B matrices used in Equation 5.2corresponding to Equation 4 are:

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -10.95 & -2.75 & 0.0043 \\ 0 & 0 & 0 & 1 \\ 0 & 24.92 & 28.58 & -0.044 \end{bmatrix} \text{ and } B = \begin{bmatrix} 0 \\ 1.94 \\ 0 \\ -4.44 \end{bmatrix}. \tag{5}$$

The system is stabilized by linear state feedback of the form $\dot{x} = (A + BK)x$. The control input, $u = Kx$ is the armature voltage of a DC-motor (V_a) and is constrained between $[-4.95, 4.95]$ volts. Additionally, this control saturation prevents the system from being globally stable. The safety controller is designed following the LMI-based Simplex approach described in Section 2. The LMI approach outputs a set of gains for the safety control K, such that when the input $u = Kx$ is used, the system will remain inside the ellipsoid also output by the method. Without saturation, the system evolves according to $\dot{x} = (A + BK)x$.

The solution to this is $x(t) = e^{(A+BK_\sigma)t}x_0$, where $x_0 \in \mathbb{R}^{4\times1}$ is an initial condition vector. Note that, as only θ and p are observable (in the control theoretic sense, but that is, are measured by sensors), $\dot{\theta}$ and $\dot{x}$ are constructed by the first-order approximations $\dot{\theta}(t) = \frac{[\theta(t)-\theta(t-mT_s)]}{mT_s}$ and $\dot{p}(t) = \frac{[p(t)-p(t-mT_s)]}{mT_s}$, where m is an integer greater than one (chosen as 2 by experimentation). In the safety and experimental controllers, this first-order approximation is accomplished by storing a buffer of previous sampled values.



(a) (b) (c)

Fig. 8: Overapproximation of the set of reachable states computed by the real-time reachability method for the linearized inverted pendulum model Equation 4 and Equation 5 for different amounts of computation runtime, 3 ms in (a), 7 ms in (b), and 16 ms in (c). The plots illustrate 2-d projections of the reachable sets for the linearized inverted pendulum from the state $x = [-0.1, 0.85, 0, 0]^T$ for reach-time 0.73. Here, the initial state is outside of the LMI-recoverable ellipsoid ($x^T P x = 1.56$), but can be proven to reenter the ellipsoid after 0.73 reach-time, despite the presence of input saturation.

5.3. Feasible and Stabilizable Regions

Next, we discuss how to compute the feasible and stabilizable regions, defined previously in Section 2.1. We use YALMIP [Löfberg 2004], the SDPT-3 [Toh et al. 1999] solver, and Matlab to solve the following semidefinite quadratic programming problem and under-approximate the recoverable states for the safety controller. For computing the stabilizable region for the safety controller, we find the gain vector during the optimization. The problem is to maximize the volume of the ellipsoid (and thus maximize the set of recoverable states) defined by:

$$\mathcal{R} = \{x \mid x^T P x \leq 1\}. \tag{6}$$

The LMI to find the positive definite P may formulated as:

$$\begin{aligned} &\min \log \det Q^{-1} \\ &\text{subject to } Q\bar{A}^T + \bar{A}^T Q < 0,\ Q > 0,\ \alpha_k^T Q a_k \leq 1,\ k = 1, \ldots, n, \end{aligned}$$

where $\bar{A} = A + BK$, $Q = P^{-1}$, and the α_k for $k \in \{1, ..., n\}$ encode the state and control constraints. Full details of this process are given in Appendix A2 of the LMI-Simplex technical report [Seto and Sha 1999].

Variants of this process may either take a given gain vector K or find a gain vector K [Seto and Sha 1999]. For our use, the output of this process is both the gain vector K and the matrix P defining a subset of the recoverable states $\mathcal{R}$, such that when the gain matrix is used for the safety controller, and the state is in $\mathcal{R}$, the state is guaranteed to stay in $\mathcal{R}$ indefinitely (since $V(x) = x^T P x$ is a Lyapunov function). Furthermore, all the constraints (including saturation limits) are satisfied for all states in $\mathcal{R}$. The gain vector K produced for the described pendulum system is $[0.4072, 7.2373, 18.6269, 3.6725]$.

5.4. Real-Time Reachability Design

In order to make use of the real-time reachability algorithm described in Section 4, the user must provide a function that minimizes / maximizes the derivative in an arbitrary box of the state space. The model used of the system is the version described above, linearized about the origin. Thus, the system dynamics are $\dot{x} = Ax + Bu$. When computing reachability for the safety controller, the gain vector K computed using the LMI approach is used, and $u = Kx$. However, due to saturation, u is limited to be in the range $[-4.95, 4.95]$ volts.

An alternative unified design could make use of the nonlinear pendulum model from Section 5.1, since the described reachability algorithm is not limited to linear systems. An advantage of such a design would be that it would permit the system state to go outside of the linearization region (in our formulation with the LMI, the recoverable region of Definition 2 specified in Lemma 5). It would be interesting as a future investigation to see how much more could be gained by allowing states outside of the linearization region, although such a gain probably strongly depends on the system being analyzed and the size of the linearization region. For our purposes, Lyapunov's indirect method ensures all states within the LMI ellipsoids are locally asymptotically stable. We recall that roughly Lyapunov's indirect method states that a nonlinear system is locally (in a neighborhood of the equilibrium point) asymptotically stable if its lineraization about an equilibrium point is globally asymptotically stable [Khalil 2002]. The bound specified in the proof of Lyapunov's indirect method gives a conservative underapproximation of the linearization region (what is typically called the domain of attraction). More sophisticated piecewise linear Lyapunov functions would yield less conservative estimates of domain of attraction.

For linear systems, the minimum and maximum derivative for any box in the state space occurs at a corner of the box. Thus, it is sufficient to sample all the corners and take the minimum and maximum due to convexity and existence of optima of convex (here, linear) functions over convex sets. This will scale exponentially with the number of dimensions (in the four-dimensional model here, there are 16 corners to sample), so for larger-dimension systems it may become necessary to examine the signs of the linear matrix in order to pick out the min/max corner more efficiently. One additional complication of the linearized model is the presence of saturation. This is handled by computing the input u at each corner, and capping it at the saturation limits before computing $\dot{x} = Ax + Bu$. To summarize, for each corner of the passed-in box, $\dot{x}$ is computed, and then the minimum or maximum is taken over all the corners. The C language program that computes $\dot{x}$ for a given dimension, given a point (corner of the box), is provided in Algorithm 2.

Another function that must be provided by the user is used to determine whether a given box is contained entirely inside the recoverable region $\mathcal{R}$. This is used to check whether the final state (box) is guaranteed to be recoverable. To do this for a single point, it suffices to know the current state x and the potential matrix P that defines the recoverable ellipsoid (output by the LMI optimization), and checking $x^T Px \leq 1$. To check this condition for a box, we check every corner point of the box.

One further function provided by the user checks is, during computation, whether the reachable region contains an unsafe state. In our case, this is a state that is outside of the linearization region where the model is considered valid. Since the constraints are all linear, it suffices to check if all of the corners of each box are in the linearization region. This computation is done at runtime to prevent saving the reachable set of states. The box passed in to this function consists of the bounding box of subsequent steps of the real-time reachability algorithm, which represents the sets of states

Algorithm 2 This function returns the derivative at a given point. The min/max derivative function would compute the derivative at each corner of a passed-in box, and take the minimum/maximum. The inputs are a particular dimension `dim`, the number of total dimensions `n`, a pointer to a state vector `state`, which is an array of length `n`, and a control saturation `u_sat`. For the pendulum case study, `n` $= 4$ and `u_sat` $= 4.95$.

```c
double derivative_at_point( int dim, int n, double * state, double u_sat) {
  double rv = 0;
  double u = 0;

  // calculate the A*x part
  for ( int i = 0; i < n; ++i)
    rv += A[dim][i] * state[i];

  // calculate the B*u part, starting with u = K*x
  for ( int i = 0; i < n; ++i)
    u += K[i] * state[i];

  // account for input saturation
  if (u < -u_sat) u = -u_sat;
  else if (u > u_sat) u = u_sat;

  // B*u
  rv += B[dim] * u;
  return rv;
}
```

reachable between two time steps, say t_i and $t_i + \delta$, where δ is the `advanceReachTime` in Algorithm 1.

5.5. Comparison between Simplex with LMI and Real-Time Reachability

We now provide a comparison between control based on the $\mathcal{R}$ from the LMI approach above, and the switching condition produced by the proposed unified approach that uses real-time reachability. For real-time reachability, we implemented the algorithm from Section 4. In order to be usable in a real-time control system, our implementation was written in C and had no dynamic memory allocations or recursion, and used no nonstandard external libraries. In our implementation, we would call the real-time reachability C code from within Matlab on either Linux and Windows. For the experiments described here, we first used a modern laptop with a quad-core Intel Core i7-2800MQ processor and 32GB RAM (although the computation does not require significant memory as described earlier). Next, we additionally evaluated the methods on embedded platforms. The first embedded platform is a BeagleBone Black development board with a 1GHz ARM processor and 512MB RAM running Debian Linux with the Xenomai real-time Linux extensions. The second embedded platform is an Arduino Yun, which has both a 400MHz MIPS processor and a 16MHz 8-bit Atmel AVR ATmega32u4 processor, and we used the ATmega32u4 for our evaluation, in part to validate our claims on minimal resources requirements. Together, these evaluations validate our claims that the real-time reachability method is cross-platform and requires minimal processing resources. The effort to port from the original x86 implementation [Bak et al. 2014] to both the ARM and AVR implementations took about two weeks of development time, which from a systems development standpoint is minimal given the insurmountable difficulties that would exist in porting all the libraries required in other existing hybrid systems reachability approaches.

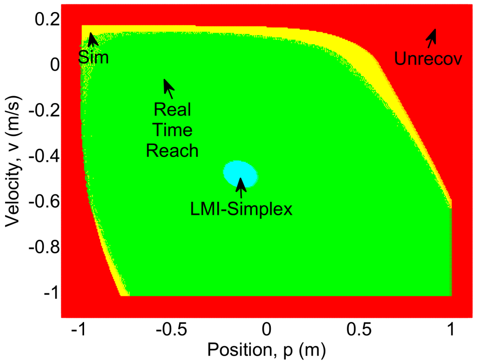


Fig. 9: Estimated projections are shown of the LMI-Simplex recoverable region $\mathcal{R}$ (cyan center set), real-time reachability recoverable region (green middle set), Simulink/Stateflow simulations that converge (yellow middle set), and simulations that diverge (red exterior set) where $\theta = 0.19$ rad (~$10.89°$) and $\dot{\theta} = 0.18$ rad (~$10.31°$) per second.

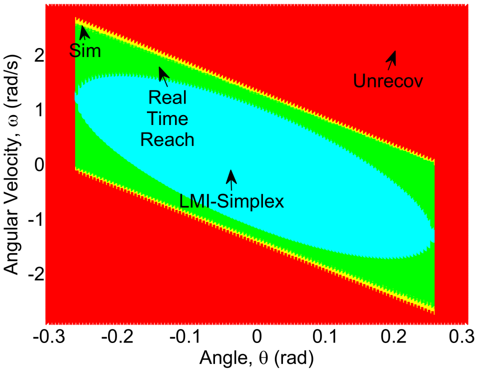


Fig. 10: Estimation of LMI-Simplex recoverable region $\mathcal{R}$ (cyan center set), real-time reachability recoverable region (green interior set), Simulink/Stateflow simulations that converge (yellow middle set), and simulations that diverge (red exterior set) shown on the projection of θ and $\dot{\theta} = \omega$ onto the $p = 0$ m and $v = 0$ m/s plane.

One remaining input for the algorithm is the reach-time necessary for a specific state to reenter $\mathcal{R}$ (the time $\delta+\alpha$ from Theorem 6). This was approximated using Euler-based simulation, which added a fixed overhead at the start of the computation. For states slightly outside of $\mathcal{R}$, the necessary reach-time was typically in the hundreds of milliseconds. Since the reachability computation incurs error due to overapproximation, we compute the set of reachable states for slightly more (1.2 times) than the time the simulation took to reach $\mathcal{R}$. If the Euler simulation did not enter $\mathcal{R}$ by some upper bound (4 seconds reach-time), the state was considered unrecoverable. A projection of the computed overapproximation of the set of reachable states for various runtimes is shown in Figure 8. As more computation runtime is added, the accuracy increases, as indicated by the size of the set decreasing.

One difference between the approaches is that the LMI-Simplex method needs to reason about one-step reachability of the plant state for any complex/smart controller command in order to compute the distance d in Figure 2. The proposed online approach, in contrast, knows what complex-controller command will be applied and can use that as part of the reachability computation. For this reason, we restrict the comparison to only examine the recoverable region for the safety controller. In this way, we do not give our approach the advantage of knowing exactly what command the complex/smart controller is using.

Our comparison shows three different approaches for estimating the recoverable region (Figures 9 and 10). First, using the LMI-only Simplex we get a subset of the recoverable region $\mathcal{R}$. Next, using a simulation-based analysis in Matlab, we can see an approximation of all recoverable states, which would be an ideal switching set. If the simulation returns to a steady state then the initial point is marked as existing in the recoverable set. Finally, we show the states that the real-time reachability-based approach can guarantee as recoverable, which is somewhere between the previous two regions. For these experiments, in order to be runnable in the control loop, the runtime for the reachability code was capped at 20 ms.

The stabilizable regions for p and $\dot{p}$ of the controller is seen in Figure 9 and the regions for θ and $\dot{\theta}$ of each controller are in Figure 10. One reason why the runtime reachability approach can recover more states is that the recoverable set contains states where input saturation occurs, whereas the set $\mathcal{R}$ contains no such states. The largest improvements in the switching set for the real-time approach occur under this saturation situation, because reachability is able to reason about the behavior of the saturated system. Another reason for the improvement is that the LMI-produced switching set must be an ellipsoid, whereas the true set of recoverable states can be an arbitrary (even non-convex) shape. This is seen in Figure 9, where, since the projection is near the maximum values of θ and $\dot{\theta}$, the LMI ellipsoid projected onto this plane is small. In Figure 10 the LMI-Simplex recoverable region is clearly ellipsoidal (as expected from Equation 6). In both Figures 9 and 10, the benefit of using real-time reachability is highlighted by the larger provably safe recoverable region. In both cases, even for a 20 ms runtime, the set of states proven recoverable using real-time reachability is very close to the simulations that converge, which means that the real-time reachability is close to optimal in estimating the actual recoverable region.

Next, we evaluated the effect of varying the runtime in real-time reachability method on the resultant switching set, which is summarized in Table I. For this table, we sampled the state-space uniformly between the state bounds presented earlier using 15 points in each dimension (so $15^4 = 50625$ points) in the hyper-rectangle $-1.25 \leq p \leq 1.25$ (m), $-1.2 \leq \dot{p} \leq 1.2$ (m/s), $-20 \leq \theta \leq 20$ (degrees), and $-30 \leq \dot{\theta} \leq 30$ (degrees/s). The columns LMI, Real-Time, Sim, and Unrecov list the number of recoverable points for each approach (in terms of recoverable states, notice that LMI $\subseteq$ RealTime $\subseteq$ Sim), as measured by the uniform sampling. The column Recoverable is the comparison of the number of states verified safe in the proposed unified method with real-time reachability over the earlier LMI-Simplex approach. The improvement is an estimate of the increased state-space size (volume) allowed using our real-time reachability method, over using only the LMI-based recoverable region. Since the real-time recoverable states contain all the LMI-Simplex states, the improvement is calculated as: $(\#\text{RealTime Points} + \#\text{LMI Points})/(\#\text{LMI Points})$. For a runtime of 20 ms, the improvement in volume of the switching set is estimated at 227%, whereas based on simulations we estimate the maximum possible improvement in Recoverable to be around 247% (calculated as $(\#\text{Sim Points} + \#\text{RealTime Points} + \#\text{LMI Points})/(\#\text{LMI Points})$).

We experimented with increasing the number of samples up to 30 points in each dimension, which yielded similar improvements, and in the limit as the number of samples tends to infinity, we would converge to the exact improvement. However, these approximations are reasonable based on the consistency of our experimental results (e.g., 20 ms runtime for 15 samples is about a 227% improvement, and it is also about a 230% improvement for 30 samples). As expected, as the runtime allowed for real-time reachability increases, the improvement increases since the real-time reachability implementation uses an anytime approach and refines the precision of the reachability computation based on available runtime. Even for small runtimes (e.g., 5ms), the improvement is already significant at over 200% more provably recoverable states, which makes the approach promising for implementation in real-time control loops.

5.6. Comparison on ARM and Arduino AVR ATmega32u4 Embedded Hardware Platforms

Next, we compare the benefit of using our real-time reachability approach versus the LMI-Simplex method on actual embedded hardware platforms. The first hardware platform is an ARM processor in the TI Sitara system-on-chip used in the CircuitCo BeagleBone Black development kit. The specific ARM processor is an AM335x 1GHz ARM Cortex-A8 with the NEON floating-point accelerator and access to 512 MB DDR3

Table I. Intel Core i7 x86-64: PC evaluation summary of experiments varying runtime.

Runtime (ms)	LMI	RealTime	Sim	Unrecov	Recoverable
5	5473	5971	14323	24858	209%
20	5473	6753	13541	24858	223%
40	5473	6974	13320	24858	227%
50	5473	7081	13213	24858	229%
75	5473	7109	13185	24858	230%
100	5473	7183	13111	24858	231%
200	5473	7273	13021	24858	233%
500	5473	7338	12956	24858	234%
1000	5473	7382	12912	24858	235%
2000	5473	7424	12870	24858	236%
3000	5473	7428	12866	24858	236%
4500	5473	7448	12846	24858	236%
6000	5473	7455	12839	24858	236%

RAM. The experiments were conducted on a Debian Linux distribution with a kernel modification to use the Xenomai real-time Linux extensions, enabling use of real-time operating system (RTOS) features within Linux. A summary of experimental results are reported in Table II. Here we can see that for reasonable runtimes even on an embedded platform (tens of milliseconds), the approach presented in this paper has an improvement of around 1.5 to 2 times over the LMI approach. For runtimes on the order of hundreds of milliseconds to seconds, the approach yields similar improvements to the desktop implementation. For this table (as with Table I), we sampled the state-space uniformly between the state bounds presented earlier using 15 points in each dimension (so $15^4 = 50625$ points) in the same hyper-rectangle used in the earlier experiment, specifically $-1.25 \leq p \leq 1.25$ (m), $-1.2 \leq \dot{p} \leq 1.2$ (m/s), $-20 \leq \theta \leq 20$ (degrees), and $-30 \leq \dot{\theta} \leq 30$ (degrees/s).

The second hardware platform is the Arduino Yun. The Yun has both a 400 MHz MIPS processor and a 16 MHz 8-bit Atmel AVR ATmega32u4. For this evaluation, we use the 16 MHz ATmega32u4 processor, which is representative of small, memory constrained embedded devices. The ATmega32u4 has available 2.5 KB SRAM, 32 KB of flash memory, but because the real-time reachability method does not use any dynamic memory allocation and does not rely on any non-standard libraries, we are able to run it on the platform in spite of the processing and memory constraints. Although the implementation runs with the restricted resources, the runtime is noticeably higher than on the with other processors. In this case, the system would only stand to benefit if the dynamics were sufficiently slow (so a runtime of seconds would be tolerable), or if we further optimized parts of the implementation for the limited resources (changing software floating-point computations to use fixed-point, since there is no FPU on the ATmega32u4. A summary of experimental results for the AVR are reported in Table III. For this table (unlike in Tables I and II), we sampled the state-space uniformly between the state bounds presented earlier using 12 points in each dimension (so $12^4 = 20736$ points) in the same hyper-rectangle as the earlier experi-

Table II. BeagleBone Black ARM: Embedded system evaluation summary of experiments varying runtime.

Runtime (ms)	LMI	RealTime	Sim	Unrecov	Recoverable
5	5473	2270	18024	24858	141%
20	5473	3832	16462	24858	170%
40	5473	4613	15681	24858	184%
50	5473	4617	15677	24858	184%
75	5473	5350	14944	24858	198%
100	5473	5361	14933	24858	199%
200	5473	5968	14326	24858	209%
500	5473	6721	13573	24858	223%
1000	5473	6952	13342	24858	227%
2000	5473	7107	13187	24858	230%
3000	5473	7110	13184	24858	230%
4500	5473	7216	13078	24858	232%
6000	5473	7216	13078	24858	232%

Table III. Arduino Atmel AVR ATmega32u4: Embedded system evaluation summary of experiments varying runtime.

Runtime (ms)	LMI	RealTime	Sim	Unrecov	Recoverable
100	2088	0	8226	10422	100%
500	2088	192	8034	10422	109%
1000	2088	566	7660	10422	127%
2000	2088	879	7347	10422	142%
3000	2088	882	7344	10422	142%
4500	2088	1198	7028	10422	157%

ments, specifically $-1.25 \leq p \leq 1.25$ (m), $-1.2 \leq \dot{p} \leq 1.2$ (m/s), $-20 \leq \theta \leq 20$ (degrees), and $-30 \leq \dot{\theta} \leq 30$ (degrees/s). While the AVR is too resource constrained to be able to improve the states usable in the control period time (of 20 ms) and requires on the order of hundreds of milliseconds to seconds to yield an improvement, this is to the best of our knowledge, the first demonstration of a reachability method in a resource constrained embedded system of this scale. We also highlight that simply performing a simulation on the AVR requires about hundreds of milliseconds of runtime.

6. RELATED WORK

The Simplex Architecture [Sha 2001; Seto and Sha 1999] has been used extensively to provide guarantees for systems that use untrusted logic. It has been used for systems ranging from off-road vehicles [Bak 2009], to models of airplanes [Seto et al. 1999], to fleets of remotely controlled cars [Crenshaw et al. 2007], to networked control systems [Yao et al. 2013]. Recently, variants of Simplex have been proposed to account for physical-system (plant) failures [Wang et al. 2013], faults in the OS or microprocessor [Bak et al. 2009], and to check for security intrusions [Mohan et al. 2013]. Simplex is closely related to Run-Time Assurance (RTA) methods [Clark et al. 2013; Murthy 2012]. RTA methods were used to construct a safe supervisory control system for a simulation of a high-altitude unmanned aerial vehicle [Aiello et al. 2010]. Here, a transition function that projects the current state to a future state was used to determine the switching boundary. This transition function as well as the recoverable states were determined through extensive simulation and online prediction of trajectories. The proposed real-time reachability approach in this work could be used to provide verified bounds on the transition function used in RTA methods. This work also mentions the interesting idea of using an online/offline design for switching module logic by leveraging a simplified model of the plant dynamics, and taking the model error into account when doing the switching, which could reduce the complexity of the online reachability computation.

Earlier work has also integrated traditionally non-real-time search approaches within real-time systems [Musliner and Durfee 1995]. In this approach, AI planning techniques were discussed in the context of real-time systems, and two categories of possible integration were proposed: 1. the non-real-time algorithms were adapted to run in a real-time fashion, or 2. they were run in a supervisory mode, not as part of the real-time control loop. Real-time reachability would fall in the former category in this classification.

A related notion to Simplex in control theory is that of a viability kernel [Aubin 1991]. A viability kernel is a set of states where there exists a trajectory that stays within a predefined environment. Viability kernels can be approximated for linear systems, for example, by using analysis of random directions in the state space [Gillula et al. 2014]. Reachability analysis of hybrid systems has also been extensively researched in the last 20 years [Guéguen et al. 2009]. Reachability analysis tools exist for classes of systems with timed [Bengtsson et al. 1996], rectangular [Henzinger et al. 1997; Frehse 2008; Johnson and Mitra 2014], linear [Frehse et al. 2011; Frehse 2008], and nonlinear [Ratschan and She 2007; Tiwari 2008; Bak 2013a; Chen et al. 2012; Benvenuti et al. 2014; Duggirala et al. 2015] dynamics, with varying degrees of accuracy and scalability. Other bounded model checking (BMC) tools for hybrid systems built on satisfiability modulo theories (SMT) solvers also exist [Eggers et al. 2011; Gao et al. 2013]. However, to the best of our knowledge, the algorithms in earlier reachability and BMC tools were all designed for offline analysis, and not for real-time, in-the-loop computation. Specifically, real-time reachability requires performance to be predictable, which is difficult to ensure when there are large external libraries, huge code bases, and signifanct use of dynamic memory. For example, one popular reachability analysis tool for affine hybrid automata is SpaceEx, which requires at least eight external libraries: Parma Polyhedra Library (PPL) [Bagnara et al. 2008], Boost C++ Libraries, GNU Multiple Precision Arithmetic Library (gmplib), GNU Linear Programming Kit (glpk), SUNDIALS (Solver Suite) [Hindmarsh et al. 2005], aaflib, ublasJama, and TinyXML [Frehse et al. 2011].[4] Another recent tool, C2E2 relies on at least eleven external libraries: GNU Linear Programming Kit (glpk and pyglpk), GNU Parser Generator (Bison), Fast Lexical Analysis (FLEX), Python, Python Parsing Libraries (Python-PLY), GTK Libraries for Python (PyGTK), Plotting Libraries for Python (Matplotlib), Packing Configurations Library (pkg-config), GNU Autoconf (autoconf), Python XML Library (lxml), and Parma Polyhedral Library (PPL).[5] While several of these libraries would not need to be executed in the reachability computation (such as those related to parsing and package management), several libraries (PPL,

[4] http://spaceex.imag.fr/licensing-45

[5] https://publish.illinois.edu/c2e2-tool/download/

gmplib, glpk, and SUNDIALS) are fundamental to the reachability computations. For these core libraries, it would be essentially impossible to convert SpaceEx or C2E2 to a real-time implementation, as several of these libraries are incredibly complex (specifically PPL, gmplib, glpk, and SUNDIALS).

The real-time reachability approach described in this paper primarily solves the problem of computing the *continuous successors* in a hybrid automaton, although it can also be applied invariant-disjoint hybrid dynamics. Research in computing continuous successors is related to validated integration, which traditionally has been done using interval analysis [Moore 1966], as well as intervals with preconditioning to reduce wrapping-effect error [Stauning 1997]. More recently, Taylor models have also been proposed as an alternative shown to provide superior long-term error control [Neher et al. 2007], and this is has been integrated into a more full hybrid automaton model checker [Chen et al. 2012]. However, the challenge for runtime approaches such as the one proposed in this paper is more with quick computation of reasonable accuracy rather than long-term error control, and we are unaware of any previous real-time validated integration approaches.

Some recent work performs online reachability computation with existing, non-real-time algorithms. This can be used, for example, when systems do not have statically-known models [Bu et al. 2011]. This work, however, treats the reachability computation as a black-box, which may or may not complete (because it does not use a real-time reachability algorithm). Another work also uses existing reachability approaches such as PHAVer [Frehse 2008] in a medical safeguard system [Li et al. 2012], and results in a system which may add safety, but only if the computation completes on time. While a theoretical upper bound on execution time may be formulated due to decidability of the particular class of hybrid automata considered [Li et al. 2014], the implementation of PHAVer does not provide such guarantees, and it is not clear that such a bound would be usable or too pessimistic. A real-time reachability algorithm that always provides an answer like our approach could be integrated into both of these approaches.

Finally, the results of formal approaches are only as good as the model they are provided. Accurate system identification [Söderström and Stoica 1988] is therefore essential. The approach here reduces pessimism in the switching logic *for a given model*. Accuracy and validation of the model itself is an important problem, but beyond the scope of this work. Recent approaches from the hybrid systems community, however, have begun made use of runtime monitors to do online checking of model accuracy [Mitsch and Platzer 2014].

7. CONCLUSION AND FUTURE WORK

In this work, we have proposed an alternate unified design for Simplex that leverages two existing design methodologies based on control-theoretic LMI optimization and hybrid systems reachability. Our unified approach extends the region where the complex/smart controller enabling smart autonomy can be used by leveraging a real-time reachability computation, and thus decreases conservatism in the switching logic. Using a runtime of 20ms (which matches the control loop period time), we were able to expand the set of states where the complex/smart controller could be used by 227%, whereas we estimated, through simulation, that the maximum improvement possible was approximately 247%. Even with a reduced real-time reachability runtime of 5ms, we were able to improve upon the LMI-based Simplex design by 213%. On embedded processors, we were also able to increase the complex/smart controller region by a factor of 1.5 to 2.0, although for an 8-bit microcontroller the current implementation was not fast enough for use at the frequency of the control loop. This improvement was demonstrated in an evaluation that uses the exact system previously used to demonstrate the LMI-based Simplex design approach, an inverted pendulum with input saturation. The real-time reachability computation is able to predict the behavior of the system despite saturation, significantly expanding the usable complex/smart controller region.

To the best of the authors' knowledge, this is the first work to present a viable real-time reachability algorithm based on the real-time systems notion of imprecise computation. The algorithm will always return an over-approximation of the set of reachable states, with better accuracy as more computation time is given. The key difference between online reachability compared with offline reachability, besides constrained runtime and resources, is that quick results are preferable to long-term error control. In our evaluation, for example, we were able to demonstrate significant improvement in the complex/smart controller region by using tens of milliseconds of computation time to bound the future behavior of the system for the next hundreds of milliseconds. Together, our evaluation on actual embedded hardware platforms including ARM processors and Atmel AVR microcontrollers illustrates the embedded usage feasibility of using the real-time reachability method. Other reachability algorithms also contain parameters which could be tuned to have some control over the computation time, such as the sampling time used in the Le Geurnic-Girard (LGG) scenario in SpaceEx [Frehse et al. 2011], and we plan to investigate better approaches for real-time reachability.

Real-time reachability has applications beyond just determining Simplex switching logic, however. We foresee future applications involving online system identification, detecting sensor spoofing, runtime verification, and enabling a variant of model-predictive control (MPC). To enable these applications, we are implementing code generation capabilities in the HYST model transformation and translation tool for hybrid automata [Bak et al. 2015], which will enable creating implementations of the real-time reachability algorithm for large classes of hybrid automata.

References

Michael Aiello, John Berryman, Jonathan Grohs, and John Schierman. 2010. Run-Time Assurance for Advanced Flight-Critical Control Systems. In *Proceedings of the American Institute of Aeronautics and Astronautics Guidance, Navigation, and Control Conference (AIAA '10)*.

R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. 1995. The algorithmic analysis of hybrid systems. *Theoretical Computer Science* 138 (1995), 3–34.

Rajeev Alur and David L. Dill. 1994. A Theory of Timed Automata. *Theoretical Computer Science* 126 (1994), 183–235.

Jean-Pierre Aubin. 1991. *Viability Theory*. Birkhauser Boston Inc., Cambridge, MA, USA.

R. Bagnara, P. M. Hill, and E. Zaffanella. 2008. The Parma Polyhedra Library: Toward a Complete Set of Numerical Abstractions for the Analysis and Verification of Hardware and Software Systems. *Science of Computer Programming* 72, 1–2 (2008), 3–21.

Stanley Bak. 2009. Industrial Application of the System-Level Simplex Architecture for Real-Time Embedded System Safety. Master's Thesis, University of Illinois at Urbana-Champaign. (2009).

Stanley Bak. 2013a. HyCreate: A Tool for Overapproximating Reachability of Hybrid Automata. (2013). http://stanleybak.com/projects/hycreate/hycreate.html

Stanley Bak. 2013b. *Verifiable COTS-based cyber-physical systems*. Ph.D. Dissertation. University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA.

Stanley Bak, Sergiy Bogomolov, and Taylor T. Johnson. 2015. HyST: A Source Transformation and Translation Tool for Hybrid Automaton Models. In *Proc. of the 18th Intl. Conf. on Hybrid Systems: Computation and Control (HSCC)*. ACM.

Stanley Bak, Deepti K. Chivukula, Olugbemiga Adekunle, Mu Sun, Marco Caccamo, and Lui Sha. 2009. The System-Level Simplex Architecture for Improved Real-Time Embedded System Safety. In *15th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS '09)*.

Stanley Bak, Ashley Greer, and Sayan Mitra. 2010. Hybrid Cyberphysical System Verification with Simplex Using Discrete Abstractions. In *Real-Time and Embedded Technology and Applications Symposium, IEEE*, Vol. 0. IEEE Computer Society, Los Alamitos, CA, USA, 143–152.

Stanley Bak, Taylor T. Johnson, Marco Caccamo, and Lui Sha. 2014. Real-Time Reachability for Verified Simplex Design. In *IEEE Real-Time Systems Symposium (RTSS)*. IEEE Computer Society, Rome, Italy.

Stanley Bak, Karthik Manamcheri, Sayan Mitra, and Marco Caccamo. 2011. Sandboxing Controllers for Cyber-Physical Systems. In *Proceedings of International Conference on Cyber-physical systems (ICCPS 2011)*.

Johan Bengtsson, Kim Larsen, Fredrik Larsson, Paul Pettersson, and Wang Yi. 1996. UPPAAL: A tool suite for automatic verification of real-time systems. In *Hybrid Systems III*, Rajeev Alur, Thomas Henzinger, and Eduardo Sontag (Eds.). LNCS, Vol. 1066. Springer, 232–243.

Luca Benvenuti, Davide Bresolin, Pieter Collins, Alberto Ferrari, Luca Geretti, and Tiziano Villa. 2014. Assume-guarantee verification of nonlinear hybrid systems with Ariadne. *International Journal of Robust and Nonlinear Control* 24, 4 (2014), 699–724.

S. Boyd, L. El Ghaoui, E. Feron, and V. Balakrishnan. 1994. *Linear Matrix Inequalities in System and Control Theory*. Studies in Applied Mathematics, Vol. 15. SIAM, Philadelphia, PA.

M.S. Branicky. 1998. Multiple Lyapunov functions and other analysis tools for switched and hybrid systems. *Automatic Control, IEEE Transactions on* 43, 4 (Apr 1998), 475–482.

Lei Bu, Qixin Wang, Xin Chen, Linzhang Wang, Tian Zhang, Jianhua Zhao, and Xuandong Li. 2011. Toward Online Hybrid Systems Model Checking of Cyber-physical Systems' Time-bounded Short-run Behavior. *SIGBED Rev.* 8, 2 (June 2011), 7–10.

Xin Chen, Erika Abraham, and Sriram Sankaranarayanan. 2012. Taylor Model Flowpipe Construction for Non-linear Hybrid Systems. *2013 IEEE 34th Real-Time Systems Symposium* 0 (2012), 183–192.

Matthew Clark, Xenofon Koutsoukos, Ratnesh Kumar, Insup Lee, George Pappas, Lee Pike, Joseph Porter, and Oleg Sokolsky. 2013. Study on Run Time Assurance for Complex Cyber Physical Systems. Technical Report, Air Force Research Lab, Wright-Patterson AFB. (2013).

Tanya L. Crenshaw, Elsa Gunter, C. L. Robinson, Lui Sha, and P. R. Kumar. 2007. The Simplex Reference Model: Limiting Fault-Propagation Due to Unreliable Components in Cyber-Physical System Architectures. In *RTSS '07*. Washington, DC, USA, 400–412.

Thao Dang. 2000. *Verification et synthese des systemes hybrides*. Ph.D. Dissertation. INPG.

Thao Dang and Oded Maler. 1998. Reachability Analysis via Face Lifting. In *Hybrid Systems: Computation and Control HSCC'98*. Springer, 96–109. LNCS 1386.

Thao Dang, Oded Maler, and Romain Testylier. 2010. Accurate hybridization of nonlinear systems. In *Proceedings of the 13th ACM international conference on Hybrid systems: computation and control (HSCC '10)*. ACM, New York, NY, USA, 11–20.

ParasaraSridhar Duggirala, Sayan Mitra, Mahesh Viswanathan, and Matthew Potok. 2015. C2E2: A Verification Tool for Stateflow Models. In *Tools and Algorithms for the Construction and Analysis of Systems*, Christel Baier and Cesare Tinelli (Eds.). Lecture Notes in Computer Science, Vol. 9035. Springer Berlin Heidelberg, 68–82.

Parasara Sridhar Duggirala, Sayan Mitra, and Mahesh Viswanathan. 2013. Verification of Annotated Models from Executions. In *Proceedings of the Eleventh ACM International Conference on Embedded Software (EMSOFT '13)*. IEEE Press, Piscataway, NJ, USA, Article 26, 10 pages.

Andreas Eggers, Nacim Ramdani, Nedialko Nedialkov, and Martin Fränzle. 2011. Improving SAT Modulo ODE for Hybrid Systems Analysis by Combining Different Enclosure Methods. In *Software Engineering and Formal Methods*, Gilles Barthe, Alberto Pardo, and Gerardo Schneider (Eds.). Lecture Notes in Computer Science, Vol. 7041. Springer Berlin / Heidelberg, 172–187.

Goran Frehse. 2008. PHAVer: Algorithmic verification of hybrid systems past HyTech. *International Journal on Software Tools for Technology Transfer (STTT)* 10 (2008), 263–279. Issue 3.

Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. 2011. SpaceEx: Scalable Verification of Hybrid Systems. In *Computer Aided Verification (CAV) (LNCS)*. Springer.

Sicun Gao, Soonho Kong, and Edmund Clarke. 2013. Satisfiability Modulo ODEs. In *International Conference on Formal Methods in Computer-Aided Design (FMCAD)*.

Jeremy H. Gillula, Shahab Kaynama, and Claire J. Tomlin. 2014. Sampling-based Approximation of the Viability Kernel for High-dimensional Linear Sampled-data Systems. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control (HSCC '14)*. ACM, New York, NY, USA, 173–182.

Hervé Guéguen, Marie-Anne Lefebvre, Janan Zaytoon, and Othman Nasri. 2009. Safety verification and reachability analysis for hybrid systems. *Annual Reviews in Control* 33, 1 (2009), 25–36.

Thomas A. Henzinger, Pei-Hsin Ho, and Howard Wong-Toi. 1997. HyTech: A model checker for hybrid systems. *Journal on Software Tools for Technology Transfer* 1 (1997), 110–122. Issue 1.

Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. 1995. What's Decidable about Hybrid Automata?. In *Journal of Computer and System Sciences*. ACM Press, 373–382.

Alan C. Hindmarsh, Peter N. Brown, Keith E. Grant, Steven L. Lee, Radu Serban, Dan E. Shumaker, and Carol S. Woodward. 2005. SUNDIALS: Suite of Nonlinear and Differential/Algebraic Equation Solvers. *ACM Trans. Math. Softw.* 31, 3 (Sept. 2005), 363–396.

Taylor T. Johnson and Sayan Mitra. 2014. Anonymized Reachability of Rectangular Hybrid Automata Networks. In *Formal Modeling and Analysis of Timed Systems (FORMATS)*.

J. Kapinski and B. H. Krogh. 2002. A new tool for verifying computer controlled systems. In *IEEE Conference on Computer-Aided Control System Design*. 98–103.

H. K. Khalil. 2002. *Nonlinear Systems* (third ed.). Prentice Hall, Upper Saddle River, NJ.

Gerardo Lafferriere, George J. Pappas, and Shankar Sastry. 2000. O-Minimal Hybrid Systems. *Mathematics of Control, Signals and Systems* 13, 1 (2000), 1–21.

J. Löfberg. 2004. YALMIP : A Toolbox for Modeling and Optimization in MATLAB. In *Proceedings of the CACSD Conference*. Taipei, Taiwan. http://users.isy.liu.se/johanl/yalmip/

Tao Li, Feng Tan, Qixin Wang, Lei Bu, Jian-Nong Cao, and Xue Liu. 2012. From Offline toward Real-Time: A Hybrid Systems Model Checking and CPS Co-design Approach for Medical Device Plug-and-Play (MDPnP). In *2012 IEEE/ACM Third International Conference on Cyber-Physical Systems (ICCPS)*. 13–22.

Tao Li, Feng Tan, Qixin Wang, Lei Bu, Jian-Nong Cao, and Xue Liu. 2014. From Offline toward Real Time: A Hybrid Systems Model Checking and CPS Codesign Approach for Medical Device Plug-and-Play Collaborations. *Parallel and Distributed Systems, IEEE Transactions on* 25, 3 (March 2014), 642–652.

Kwei-Jay Lin, Swaminathan Natarajan, and Jane W.-S. Liu. 1987. Imprecise Results: Utilizing Partial Computations in Real-Time Systems. In *RTSS*. 210–217.

C.L. Liu and J.W. Layland. 1973. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *Journal of the Association for Computing Machinery* 20, 1 (1973).

J. W S Liu, Wei-Kuan Shih, Kwei-Jay Lin, R. Bettati, and J. Y Chung. 1994. Imprecise computations. *Proc. IEEE* 82, 1 (Jan 1994), 83–94.

Stefan Mitsch and Andre Platzer. 2014. ModelPlex: Verified Runtime Validation of Verified Cyber-Physical System Models. In *Runtime Verification*, Borzoo Bonakdarpour and Scott A. Smolka (Eds.). Lecture Notes in Computer Science, Vol. 8734. Springer International Publishing, 199–214.

Sibin Mohan, Stanley Bak, Emiliano Betti, Heechul Yun, Lui Sha, and Marco Caccamo. 2013. S3A: Secure System Simplex Architecture for Enhanced Security and Robustness of Cyber-physical Systems. In *Proceedings of the 2Nd ACM International Conference on High Confidence Networked Systems (HiCoNS '13)*. 10.

R.E. Moore. 1966. *Interval analysis*. Prentice-Hall.

Abhishek Murthy. 2012. Simplex Architecture for Run Time Assurance of Hybrid Systems. Safe and Secure Systems and Software Symposium (S5). (2012).

David J. Musliner and Edmund H. Durfee. 1995. World Modeling for the Dynamic Construction of Real-time Control Plans. *Artif. Intell.* 74, 1 (March 1995), 83–127.

M. Neher, K. R. Jackson, and N. S. Nedialkov. 2007. On Taylor model based integration of ODEs. *SIAM J. Numer. Anal* 45 (2007).

Stefan Ratschan and Zhikun She. 2007. Safety verification of hybrid systems by constraint propagation-based abstraction refinement. *ACM Trans. Embed. Comput. Syst.* 6, 1, Article 8 (Feb. 2007).

Danbing Seto, Enrique Ferreira, and Theodore F. Marz. 1999. *Case Study: Development of a Baseline Controller for Automatic Landing of an F-16 Aircraft Using Linear Matrix Inequalities (LMIs)*. Technical Report. citeseer.ist.psu.edu/606539.html

D. Seto and Lui Sha. 1999. *A Case Study on Analytical Analysis of the Inverted Pendulum Real-Time Control System*. CMU/SEI Tech. Rep. 99-TR-023. Carnegie Mellon University, Pittsburgh, PA.

Lui Sha. 2001. Using Simplicity to Control Complexity. *IEEE Softw.* 18, 4 (2001), 20–28.

T. Söderström and P. Stoica (Eds.). 1988. *System Identification*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

O. Stauning. 1997. *Automatic validation of numerical solutions*. Ph.D. Dissertation. Informatics and Mathematical Modelling, Technical University of Denmark, DTU, Richard Petersens Plads, Building 321, DK-2800 Kgs. Lyngby.

Ashish Tiwari. 2008. Abstractions for hybrid systems. *Formal Methods in System Design* 32, 1 (2008), 57–83.

K. C. Toh, M. J. Todd, and R. H. Tutuncu. 1999. SDPT3: a MATLAB software package for semidefinite programming. *Optimization Methods and Software* 11 (1999), 545–581.

Lieven Vandenberghe, Stephen Boyd, and Shao-Po Wu. 1998. Determinant Maximization with Linear Matrix Inequality Constraints. *SIAM J. Matrix Anal. Appl.* 19, 2 (1998), 499–533.

Xiaofeng Wang, N. Hovakimyan, and Lui Sha. 2013. L1Simplex: Fault-tolerant control of cyber-physical systems. In *Cyber-Physical Systems (ICCPS), 2013 ACM/IEEE International Conference on*. 41–50.

Jianguo Yao, Xue Liu, Guchuan Zhu, and Lui Sha. 2013. NetSimplex: Controller Fault Tolerance Architecture in Networked Control Systems. *Industrial Informatics, IEEE Transactions on* 9, 1 (Feb. 2013), 346–356.

ACKNOWLEDGMENTS

The authors are grateful for the feedback from the anonymous reviewers of this paper and its earlier revision [Bak et al. 2014] that significantly helped improve this paper. The material presented in this paper is based upon work supported by the National Science Foundation (NSF) under grant numbers CNS-1302563, CNS-1219064, CNS 13-29886, CNS 13-30077, CNS 1464311, and CCF 1527398, by the Office of Naval Research (ONR) under grant number N00014-12-1-0046, by the Air Force Research Laboratory's Information Directorate (AFRL/RI) through the Visiting Faculty Research Program (VFRP) under contract number FA8750-13-2-0115, AFRL through contract number FA8750-15-1-0105, the Air Force Office of Scientific Research (AFOSR) in part under contract number FA9550-15-1-0258, and the AFOSR Summer Faculty Fellowship Program (SFFP). The U.S. government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of AFRL, AFRL/RI, AFOSR, NSF, or ONR.

Received December 14, 2014; revised June 24, 2015; accepted September 2, 2015

Quantified Bounded Model Checking for Rectangular Hybrid Automata

Luan Viet Nguyen
University of Texas at Arlington

Djordje Maksimovic
University of Toronto

Taylor T. Johnson
University of Texas at Arlington

Andreas Veneris
University of Toronto

***Abstract*—Satisfiability Modulo Theories (SMT) solvers have been successfully applied to solve many problems in formal verification such as bounded model checking (BMC) for many classes of systems from integrated circuits to cyber-physical systems (CPS). Typically, BMC is performed by checking satisfiability of a possibly long, but quantifier-free formula. However, BMC problems can naturally be encoded as quantified formulas over the number of BMC steps. In this approach, we then use decision procedures supporting quantifiers to check satisfiability of these quantified formulas. This approach has previously been applied to perform BMC using a Quantified Boolean Formula (QBF) encoding for purely discrete systems, and then discharges the QBF checks using QBF solvers. In this paper, we present a new quantified encoding of BMC for rectangular hybrid automata (RHA), which requires using more general logics due to the real (dense) time and real-valued state variables modeling continuous states. We have implemented a preliminary experimental prototype of the method using the HyST model transformation tool to generate the quantified BMC (QBMC) queries for the Z3 SMT solver. We describe experimental results on several timed and hybrid automata benchmarks, such as the Fischer and Lynch-Shavit mutual exclusion algorithms. We compare our approach to quantifier-free BMC approaches, such as those in the dReach tool that uses the dReal SMT solver, and the HyComp tool built on top of nuXmv that uses the MathSAT SMT solver. Based on our promising experimental results, QBMC may in the future be an effective analysis approach for RHA as further improvements are made in quantifier handling in SMT solvers such as Z3.**

***Index Terms*—bounded model checking, hybrid automata, timed automata, satisfiability modulo theories**

I. Introduction

Boolean Satisfiability (SAT) is the canonical NP-complete problem and is to determine if a given Boolean formula is satisfiable, i.e., check if there exists an assignment of values to variables where the formula is true. A Boolean formula is given in Conjunctive Normal Form (CNF), that is, a conjunction of clauses, each of which is a disjunction of literals. Satisfiability modulo theories (SMT) is a generalization of SAT, where literals are interpreted with respect to a background theory (e.g., linear real arithmetic, nonlinear integer arithmetic, bit-vectors, etc.).

Recently, SMT-based techniques have been developed to formally verify hybrid systems [1]–[6]. Typically, these SMT-based methods are used in bounded model checking (BMC), which is to check for a transition system A and a specification P whether $I(s_0) \wedge \bigwedge_{i=0}^{k-1} T(s_i, s_{i+1}) \wedge (\bigvee_{i=0}^{k} P(s_i))$ is satisfiable. Here, $I(s_0)$ encodes an initial set of states over a set of variables s_0, $T(s_i, s_{i+1})$ represents the transition relation from iteration i to $i+1$ over sets of variables s_i and s_{i+1}, and $P(s_i)$ encodes the specification at step i.

Hybrid automata are a modeling formalism used to verify dynamical systems including both continuous states and dynamics as well as discrete states and transitions. Examples of systems naturally modeled by hybrid automata arise in the interaction of physical plants and software controllers in real-time systems and cyber-physical systems (CPS). In essence, hybrid automata augment finite state machines with a set of real-valued variables that evolve continuously over intervals of real time. In hybrid automata, a transition relation $T = D \cup \mathcal{T}$ encodes both discrete transitions D and continuous trajectories $\mathcal{T}$ over intervals of real-time. Rectangular hybrid automata (RHA) are a special class of hybrid automata with continuous dynamics described by rectangular differential inclusions and where all other quantities (guard conditions, invariants, resets, etc.) of the automata are linear inequalities over constants [2], [7]. Sets of states, as well as discrete transitions and continuous trajectories of RHA, can be symbolically represented by SMT formulas over real and Boolean variables.

Depending on the underlying logics supported, SMT solvers may or may not support quantifiers. While quantifiers make the language more expressive, they increase the complexity of computations like checking satisfiability and may also lead to undecidability. Techniques allowing quantifiers, such as in quantified Boolean formula (QBF) solvers, have been developed for BMC of purely discrete systems, such as finite state machines [8], [9]. However, to the best of our knowledge, there has been no effort to develop quantified BMC (QBMC) methods for timed or hybrid automata, which we develop in this paper. Of course, this is partially because the underlying SMT solver requires support for complex combination theories and efficient algorithms to check quantified formulas, which until recently, were either not available or not scalable.

The logic used requires some finite sort for the discrete states (such as a enumerated type or bitvectors) and reals for the continuous states and trajectories. In this paper, we use LRABV (linear real arithmetic with bit-vectors) for encoding QBMC for timed automata and RHA, and we note that general hybrid automata would need NRABV (nonlinear real arithmetic with bit-vectors) or beyond, such as those whose solutions involve special (transcendental) functions like sin, cos, exp, etc. While none of these logics are officially supported in the SMT-LIB2 standard (nor the 2.5 draft) as of the time of this writing [10], several solvers do have unofficial support for this combination theory, such as the latest versions of Z3, which is the SMT solver used in this paper [11].

Related Work: When defining the semantics of hybrid automata, first-order or higher logic is typically used and quantifiers typically show up in several places. Existential quantifiers over reals are used to specify that some amount of real time may elapse in a given location of the hybrid automaton. Universal quantifiers over reals representing real time are used to construct invariants that are enforced at all times, while in a given location of the hybrid automaton; otherwise real time is not allowed to advance, and a discrete transition must be taken, if any are enabled based on the current state and guards of the transitions. Alternative approaches to the one described in this paper have previously been developed, where the universal quantifiers used to define invariants' semantics are explicitly removed from the SMT expressions to create quantifier-free formulas. This allows the use of existing SMT-based procedures and avoids quantifier-elimination and other quantifier-handling procedures [2], [3], [12]. We note that this approach does not use quantifiers on the number of steps $k \geq 0$ in the BMC computation. which we do in this paper. Specifically, we suggest that effectively encoding the BMC problem in a quantified form over the number of steps k may provide a more scalable approach in the future as quantifier handling procedures are improved in the underlying solvers. We accomplish this by extending existing results for BMC of discrete systems with QBF solvers [9] to timed and hybrid automata, specifically RHA.

Typical approaches to analyze timed and hybrid automata use symbolic representations of states such as difference bound matrices (DBMs) to represent clock regions in Uppaal [13] or polyhedra in HyTech [14]. Several other formal verification tools for hybrid automata focus on performing reachability computations, and overapproximate the set of reachable states using various data structures to symbolically represent geometric sets of states, such as Taylor models in Flow* [15] and support functions in SpaceEx [16]. Reachability analysis tools like Flow* and SpaceEx focus on computing reachable states, although there is a direct equivalence between time-bounded reachability computations and BMC.

Several SMT-based approaches can verify properties of timed and hybrid automata. dReal is an SMT-solver for first-order logic formulas over the reals, and uses a δ-complete decision procedure [17]. dReach is a BMC tool that queries dReal to check satisfiability of SMT formulas encoding the transitions and trajectories for hybrid automata [4]. HyComp is a verification tool for networks (parallel compositions) of hybrid automata with polynomial and other dynamics [6] and is built on top of nuXmv [18]. For k-induction and IC3, HyComp may perform unbounded model checking, but in the BMC mode, it also allows a limit on the number of steps, and also encodes the semantics of the network of hybrid automata's transition relation and trajectories. A very closely related approach to this paper also encodes BMC problems for timed automata using quantified formulas, but this quantification is to encode unknown or incomplete components, and is not a quantification over the BMC length [19]. Passel is a parameterized verification tool for networks of RHA that may prove properties regardless of the number N of automata in the network [2]. Passel implements an extension to hybrid automata of the invisible invariants approach for parameterized verification, and consists of an invariant synthesis procedure [20] that relies on reachability computations [5]. Passel encodes the semantics of networks of hybrid automata as SMT formulas and checks satisfiability and validity using the Z3 SMT solver. Additionally, when performing reachability computations, Passel makes use of quantifier elimination procedures over the reals and bit-vectors [5].

Contributions: In this paper, we present a new SMT-based verification technique that encodes the BMC problem for RHA in a quantified form, which we call quantified BMC (QBMC). We take hybrid automata in the SpaceEx format [16], which are then translated to the QBMC encoding proposed in this paper using the HyST model transformation tool [21]. We then perform QBMC by querying the Z3 SMT solver via its Python API and use its quantifier-handling procedures [11]. We present preliminary experimental results where the QBMC approach and Z3 perform competitively, when compared to (a) the dReach tool that performs BMC using an SMT check by querying the dReal δ-decidable SMT solver [4], [17], and (b) the HyComp tool built on top of nuXmv that uses the MathSAT SMT solver [22]. The examples include standard ones such as Fischer and Lynch-Shavit mutual exclusion, as well as an illustrative example to describe the encoding. The main contribution of this paper is the first encoding of BMC as a quantified problem for RHA. Our results subsume the case for timed automata, as RHA are more expressive than timed automata, and we note this is also the first QBMC approach for timed automata.

II. Hybrid Automata Syntax and Semantics

A hybrid automaton is essentially a finite state machine extended with a set of real-valued variables that evolve continuously over intervals of real-time.

Syntax: The syntactic structure of a hybrid automaton is formally defined as follows.

Definition 1: A hybrid automaton $\mathcal{H}$ is a tuple, $\mathcal{H} \triangleq \langle Loc, Var, Inv, Flow, Trans, Init \rangle$, with the components as follows. (a) Loc is a finite set of discrete locations. (b) Var is a finite set of n continuous, real-valued variables, and $\mathcal{Q} \triangleq Loc \times \mathbb{R}^n$ is the state-space. (c) Inv is a finite set of invariants, one for each discrete location, and for each location $\ell \in Loc$, $Inv(\ell) \subseteq \mathbb{R}^n$. (d) $Flow$ is a finite set of ordinary differential inclusions, one for each continuous variable $x \in Var$, and $Flow(\ell, x) \subseteq \mathbb{R}^n$ describes the continuous dynamics in each location $\ell \in Loc$. (e) $Trans$ is a finite set of transitions between locations. Each transition is a tuple $\tau \triangleq \langle \ell, \ell', g, u \rangle$, where ℓ is a source location and ℓ' is a target location that *may* be taken when a guard condition g is satisfied, and the post-state is updated by an update map u. (f) $Init$ is an initial condition, which consists of a set of locations in Loc and a formula over Var, so that $Init \subseteq \mathcal{Q}$.

For RHA, all the expressions appearing in invariants, guards, and updates must be boolean combinations of constant inequalities, and the flows are rectangular differential inclusions ($\dot{x} \in [a, b]$ for $a \leq b$) [7]. We use the dot (.) notation to refer to different components of tuples e.g., $\mathcal{H}.Inv$ refers to the invariants of automaton $\mathcal{H}$ and $\tau.g$ refers to the guard of a transition τ. If clear from context, we drop $\mathcal{H}$ and τ and refer to the individual components of the tuple.

Semantics: The semantics of a hybrid automaton $\mathcal{H}$ are defined in terms of executions, which are sequences of states. A *state* q of $\mathcal{H}$ is a tuple $q \triangleq \langle \ell, v \rangle$, where $\ell \in Loc$ is a location, and $v \in \mathbb{R}^n$ is a valuation of all variables in Var. Formally, for a set of variables Var, a *valuation* is a function mapping each $x \in Var$ to a point in its type—here, $\mathbb{R}$. The state-space $\mathcal{Q}$ is the set of all states of $\mathcal{H}$. Updates of states are described by a transition relation $T \subseteq \mathcal{Q} \times \mathcal{Q}$. For a transition $\langle q, q' \rangle \in T$ where $q \triangleq \langle \ell, v \rangle$ and $q' \triangleq \langle \ell', v' \rangle$, we denote $q \to q' \in T$ as the transition between the current state q and the next state q'. The transition relation T is partitioned into disjoint sets of discrete transitions and continuous trajectories that respectively describe the discrete and continuous behaviors of the automaton. Thus, $T \triangleq D \cup \mathcal{T}$, where: (a) $D \subseteq \mathcal{Q} \times \mathcal{Q}$ is the set of discrete transitions that describe instantaneous updates of state, (b) $\mathcal{T} \subseteq \mathcal{Q} \times \mathcal{Q}$ is the set of continuous trajectories that describe updates of state over real time intervals.

Discrete transitions. A discrete transition $q \to q' \in D$ models an instantaneous update from the current state q to the next state q'. There is a discrete transition $q \to q' \in D$ if and only if (iff): $\exists \tau \in Trans : q.v \models \tau.g \wedge q'.v' \models \tau.u$, where $\tau.g$, and $\tau.u$ are the guard condition and the update map of the discrete transition τ, respectively.

Continuous trajectories. A continuous trajectory $q \to q' \in \mathcal{T}$ models the update of state q to q' over an interval of real time. The set-valued function Δ returns a set of states and is defined as: $\Delta(q.\ell, q.v, x, t) \in q.v.x + \int_{\delta=t_0}^{t} f(q.\ell, x) d\delta$, where $f \in Flow$ is a flow rate. A formula over $Var \cup \dot{Var}$ that describes the evolution of a real variables $x \in Var$ over a real time interval $J = [t_0, t]$, and $q.v.x$ is the value of continuous variable x of the state q at $t = t_0$. Then, there is a trajectory $q \to q' \in \mathcal{T}$ iff: $\exists t_\alpha \in \mathbb{R}_{\geq 0}\ \forall t_\beta \in \mathbb{R}_{\geq 0}\ \exists \ell \in Loc : t_\beta \leq t_\alpha \ \wedge\ \Delta(q.\ell, q.v, Var, t_\beta) \models Inv(\ell) \ \wedge\ q'.v'.Var \in \Delta(q.\ell, q.v, Var, t_\alpha)$. For each real variable x, $q.v.x$ must evolve to the valuation $q'.v'.x$ at precisely time t_α and corresponding to the flow rate of x in location ℓ. Additionally, all states along the trajectory must satisfy the invariant $Inv(\ell)$ i.e., at every point in the interval of real time $t_\beta \leq t_\alpha$.

Executions. An *execution* of $\mathcal{H}$ is a sequence $\pi \triangleq q_0 \to q_1 \to q_2 \to$ ..., such that: (a) $q_0 \in Init$ is an initial state, and (b) either $q_i \to q_{i+1} \in D$ is a discrete transition or $q_i \to q_{i+1} \in \mathcal{T}$ is a continuous trajectory for each consecutive pair of states in the sequence π. A state $q_k \triangleq \langle \ell_k, v_k \rangle$ is *reachable* from initial state $q_0 \triangleq \langle \ell_0, v_0 \rangle \in Init$ iff there exists a finite execution $\pi \triangleq q_0 \to q_1 \to ... \to q_k$.

Safety specifications. In this paper, we develop the QBMC procedure to check whether safety properties of hybrid automata are satisfied up-to iteration k. A *safety specification* ϕ is a formula over Loc and Var that describes a set of states $[\![\phi]\!] \subseteq \mathcal{Q}$, where $[\![\cdot]\!]$ is the set of states satisfying ϕ. For an automaton $\mathcal{H}$ and a safety specification ϕ, the automaton satisfies the specification, denoted $\mathcal{H} \models \phi$, iff for every execution π, for every state $q_0, q_1, \ldots, q_k$ in the execution π, we have $\pi.q_k \in [\![\phi]\!]$. If $\mathcal{H} \models \phi$ for every $i \in \{0, \ldots, k\}$, then the system is safe up-to iteration k. If $\mathcal{H} \models \phi$ for any k, then the system is safe. For a safety specification ϕ, a *counterexample* is an execution π where some state $q \in \pi$ violates ϕ, i.e., $q \not\models \phi$, or equivalently, $q \notin [\![\phi]\!]$.

III. Quantified BMC for Hybrid Automata

Bounded model checking (BMC) has been used widely in verification and falsification of safety and liveness properties of various classes of systems, from finite state machines to hybrid automata. The key idea is to search for a counterexample execution whose length is bounded by a number of steps k. In other words, BMC will explore all executions from any initial state of the system Ψ to detect whether there is a way to reach a bad state that violates a given property (or to find a loop in the case of liveness). Then this path is considered as a counterexample to the property that may help the user to debug the system. For finite state systems, BMC can be encoded as a propositional formula to be checked as satisfiable or unsatisfiable using a Boolean SAT solver. For hybrid automata, BMC can be encoded as a formula over reals and finite sorts (such as Booleans, bitvectors, or enumerated types). In this paper, we focus only on hybrid automata with rectangular differential inclusion dynamics ($\dot{x} \in [a, b]$ for real constants $a \leq b$), and for this class of automata, the formulas are within linear real arithmetic (LRA). We first illustrate BMC for hybrid automata using the traditional quantifier-free encoding, and then describe the quantified BMC (QBMC), which is the main contribution of this paper.

Quantifier-Free BMC for Hybrid Automata: Let P be a set of given specifications of the hybrid automata, the BMC problem will determine whether a specification $P(q_k) \in P$ is safe after k steps, and it is:

$$\Phi(k) \triangleq I(V_0) \wedge \bigwedge_{i=0}^{k-1} T_i(V, V') \wedge (\bigvee_{i=0}^{k} P(V_i)), \qquad (1)$$

where V_i corresponds to the set of variables Var of the automaton $\mathcal{H}$ appropriately renamed. For example, V_i contains of every variable $v \in Var$ syntactically renamed to v_i, etc., and V' consists of primed variables, e.g., v' for each $v \in Var$. In Equation 1, $I(V_0)$ encodes the initial set of states, $T_i(V, V')$ encodes the transition between consecutive pairs of sets of states, and $P(V_i)$ is a safety specification at iteration i. We note that the sets of variables V_i for each iteration i are implicitly existentially quantified and e.g., we could equivalently prefix $\exists V_0, V_1, \ldots, V_k$. We drop the sets of variables for a shorter notation, e.g., Equation 1 is equivalent to $I_0 \wedge \bigwedge_{i=0}^{k-1} T_i \wedge (\bigvee_{i=0}^{k} P_i)$.

Example 1: Consider the hybrid automaton $\mathcal{H}$ shown in Figure 1. Assume that the automaton starts at location ℓoc_1, and the initial value of x is 0. The set of bad states are defined by: $P \triangleq \bigvee_{i=0}^{k} \neg(q_i.\ell_i = \ell oc_2 \implies x \geq 2.5)$. Two intervals $[a_1, b_1]$ and $[a_2, b_2]$ describe the rectangular differential inclusions for locations ℓoc_1, and ℓoc_2, respectively. This automaton would be a *timed automaton* if all of the constants values are equal, i.e., $a_1 = b_1 = a_2 = b_2$. This automaton would be a *multi-rate timed automaton* if $a_1 = b_1$ and $a_2 = b_2$ but possibly $a_1 \neq a_2$. Otherwise, this automaton is a *rectangular hybrid automaton*. Suppose that $a_1 = 1$, $b_1 = 2$, $a_2 = 3$, and $b_2 = 4$. We introduce $k + 1$ copies $x_0, x_1, ..., x_k$ and $\ell_0, \ell_1, ..., \ell_k$, where the variable x_i gives the value of the variable x, and ℓ_i indicates the location at the state q_i, representing the i^{th} step of the BMC computation for the automaton shown in Figure 1. The BMC computation of $\mathcal{H}$ for each k up to 2 can be encoded as:

- $k = 0$: $I_0 := (\ell_0 = \ell oc_1 \wedge x_0 = 0)$;
- $k = 1$ (D_0): $(\ell_0 = \ell oc_1 \wedge \ell_1 = \ell oc_2 \wedge x_0 \leq 5 \wedge x_0 \geq 2.5 \wedge x_1 = x_0)$,
- $k = 1$ $(\mathcal{T}_0)$: $(\ell_0 = \ell oc_1 \implies (\ell_1 = \ell_0 \wedge x_0 + a_1\delta \leq x_1 \wedge x_1 \leq x_0 + b_1\delta \wedge x_1 \leq 5))$,
- $k = 2$ (D_1): $(\ell_1 = \ell oc_1 \wedge \ell_2 = \ell oc_2 \wedge x_1 \leq 5 \wedge x_1 \geq 2.5 \wedge x_2 = x_1)$,
- $k = 2$ $(\mathcal{T}_1)$: $(\ell_1 = \ell oc_1 \implies (\ell_2 = \ell_1 \wedge x_1 + a_1\delta \leq x_2 \wedge x_2 \leq x_1 + b_1\delta \wedge x_2 \leq 5))$,

where δ is a fresh, real constant.[1] We split the discrete transitions and trajectories for clarity, but the entire formula to be checked for iteration $k = 1$ would just be the disjunction of these conjuncted with the formula representing $k = 0$ and the bad set of states, i.e., $I_0 \wedge (D_0 \vee \mathcal{T}_0) \wedge P$. For $k = 2$, this full formula would be $I_0 \wedge (D_0 \vee \mathcal{T}_0) \wedge (D_1 \vee \mathcal{T}_1) \wedge P$.

For $k = 1$, we dropped the obviously infeasible transition from ℓoc_2 to ℓoc_1 from D_0, which would be found as being unsatisfiable since $\ell_0 \neq \ell oc_2$. However, the transition from ℓoc_1 to ℓoc_2 also cannot occur since $x_0 = 0$, but $x_0 \not\geq 2.5$, so that part is unsatisfiable and no discrete transitions may be taken from the set of initial states. We also dropped the continuous dynamics for ℓoc_2 from $\mathcal{T}_0$ since this would also be infeasible since $\ell_0 \neq \ell oc_2$. However, real time may elapse, and as encoded, would correspond to any choice of time δ such that $x_1 \in [a_1\delta, b_1\delta]$ and $x_1 \leq 5$. Since $a_1 = 1$ and $b_1 = 2$, at most between 2.5 and 5 seconds of real time could elapse, and either case would yield $x_1 \in [0, 5]$.

For $k = 2$, we also dropped the infeasible transition and trajectory for clarity. In this case, the transition from ℓoc_1 to ℓoc_2 is enabled since $x_1 \in [0, 5]$, so the update to ℓoc_2 may occur. However, now the continuous trajectory would be infeasible since x_1 could already be 5 and the invariant requires $x_2 \leq 5$, so no real-time $\delta > 0$ may elapse, as otherwise $x_1 + a_1\delta > 5$ is unsatisfiable for $x_1 = 5$. So, the only state update would be to ℓoc_2 owing to the discrete transition.

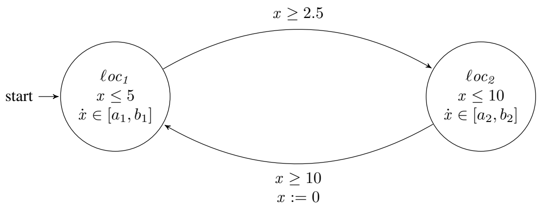


Fig. 1. The hybrid automaton $\mathcal{H}$ for Example 1.

Quantified BMC (QBMC) for Hybrid Automata: Next, we construct a quantified formula $\Omega(k)$ for BMC of $\mathcal{H}$ of length k. We introduce a vector $t = \langle t_1, t_2, ..., t_{\lceil \log_2 k \rceil} \rangle$ to index each iteration of the BMC of $\mathcal{H}$. The current state q and next state q' under the transition relation $T(V, V')$ are connected to the current state and the next state for each particular iteration t_i, for $i \in [1, \lceil \log_2 k \rceil]$. The quantified BMC formula is:

$$\Omega(k) \triangleq \exists V_0, V_1, ..., V_k, \delta \forall t \exists V, V' \mid I(V_0) \wedge T(V, V') \wedge \bigwedge_{i=0}^{k-1} t_{i+1} \rightarrow [(V = V_i) \wedge (V' = V_{i+1})] \wedge (\bigvee_{i=0}^{k} P(V_i)),$$

where we note that the existential δ encodes the real time elapse and would appear in the trajectories $\mathcal{T}$ of the disjunct $T = D \vee \mathcal{T}$.

For $k = 3$, the QBMC of the automaton of Example 1 is:

$$\begin{aligned} \Omega(3) = \exists V_0, V_1, V_2, V_3, \delta \forall t_1, t_2 \exists V, V' \mid I(V_0) \wedge T(V, V') \\ \wedge \ \{\bar{t_1} \rightarrow [(V = V_0) \wedge (V' = V_1)]\} \\ \wedge \ \{t_1 \wedge \bar{t_2} \rightarrow [(V = V_1) \wedge (V' = V_2)]\} \\ \wedge \ \{t_1 \wedge t_2 \rightarrow [(V = V_2) \ \wedge (V' = V_3)]\} \\ \wedge \ (P(V_0) \ \vee P(V_1) \ \vee P(V_2) \ \vee P(V_3)), \end{aligned} \quad (2)$$

where $V = V'$ is a shorthand indicating every variable $v \in V$ equals its corresponding counterpart $v' \in V$. In Equation 2, if the value of t_1 is 0, then there is a continuous trajectory that evolves from the initial state q_0, where $q_0.\ell_0 = \ell oc_1$ and $x_0 = 0$, to the next state q_1, where $q_1.\ell_1 = \ell oc_1$ and $x_1 \leq 5$. When $t_1 = 1$ and $t_2 = 0$, the system takes the discrete transition from the current state q_1 to the next state q_2, where $q_2.\ell_2 = \ell oc_2$ and the value of x_3 is not higher than 10. At $k = 3$, both t_1, and t_2 are true, then q_2 becomes the current state, and q_3 is the next state, where $q_3.\ell_3 = \ell oc_1$, and $x_3 \leq 5$. The discrete transition taken from q_2 to q_3 when $x \geq 10$ will reset the value of x to 0.

If it terminates, an SMT solver supporting the combined theory of bitvectors and reals with quantifiers will return SAT for the QBMC formula iff there exists an execution from an initial state to a bad state, i.e., if a bad state is reachable. Otherwise, if it terminates, it will return UNSAT if a bad state is not reachable in k steps. We note that the combination theory of linear real arithmetic with bitvectors is decidable, and Z3 is in essence a decision procedure for this theory.

IV. Experimental Results

We implement the method described in this paper as a module within HyST [21]. HyST takes as input a hybrid automaton model in an extended form of the SpaceEx XML format [16] (supporting e.g., nonlinear functions instead of only affine ones), and creates the transition relation as SMT

[1] In general, a universally quantified assertion that the invariant is satisfied for every real time along the trajectory from time t_0 to time $t_0 + \delta$, although this is unnecessary for rectangular differential inclusions with linear guards and invariants for convexity reasons [2], [6], which makes this assertion fall into the combination theory of linear real arithmetic with bitvectors (or some finite sort to encode the locations).

TABLE I
EXAMPLE 1 PERFORMANCE COMPARISON.

Tools	L	k ≤ 32		k ≤ 64		k ≤ 128	
		Time (sec)	Mem (MB)	Time (sec)	Mem (MB)	Time (sec)	Mem (MB)
QBMC	2	1.11	27.2	3.68	39.4	19.9	91.2
dReach	2	86.7	102.4	1176.4	284.7	20034	829.2
HyComp	2	0.4	97.3	0.6	101.8	1.44	109.3

formulas using the Z3 Python API. We evaluate the QBMC method described in this paper on several examples.[2] We compare the results from the QBMC method of this paper with that of dReach, which is a state-of-the-art BMC tool for nonlinear hybrid automata [23], and with that of HyComp that uses the MathSAT SMT solver [6]. All of the models for dReach and HyComp are also generated using HyST. The experiments are performed on Intel I5 2.4GHz processor with 3GB RAM, executing the method described in this paper and dReach in a VirtualBox virtual machine running Ubuntu 64-bit. Z3 version 4.3.2 was used in the evaluation. We collect the running times (Time) in seconds and the peak memory usages (Mem) in megabytes for different examples.

We first evaluate our QBMC encoding on the illustrative hybrid automata presented in Example 1, and compare the results to those of dReach and HyComp. The performances of those three different methods are shown in Table I, where QBMC denotes the QBMC presented in this paper, k is a number of steps in the BMC computation, and L is the number of discrete locations. The constants values are given as: $a_1 = 0, b_1 = 1, a_2 = 0$, and $b_2 = 2$. The results shown in Table I preliminarily indicate that our QBMC approach is capable of solving BMC significant faster than dReach, but slower than HyComp. However, our approach requires less memory usage compared to dReach and HyComp.

Next, we evaluate QBMC with several scenarios using the Fischer mutual exclusion protocol [2]. Fischer mutual exclusion is a timed distributed algorithm that ensures a mutual exclusion safety property, namely that at most one process in a network of N processes may enter a critical section simultaneously. The set of bad states is defined by:

$$\phi \triangleq \neg\forall i,j \in \{1,\ldots,N\} \mid (i \neq j \wedge \ q_i = cs) \ \rightarrow \ q_j \neq cs,$$

where q_i and q_j are variables modeling the discrete location of the automata, cs is the critical section location, and $\rightarrow$ is logical implication. We compare the performance of QBMC in solving the BMC of Fischer protocol with HyComp and dReach. Figures 2 and 3 show, respectively, the runtime and memory usage comparison among HyComp, dReach and QBMC for different numbers of processes of Fischer protocol; where QBMC-safe, QBMC-unsafe, HyComp-safe, HyComp-unsafe, dReach-safe, and dReach-unsafe denote the BMC of the safe and unsafe version of Fischer protocol using QBMC, HyComp, and dReach, respectively. Overall, HyComp is generally faster than QBMC. However, it requires a higher memory consumption than QBMC. For instance, with $k \leq 16$, the BMC of the unsafe version of Fischer protocol with 5 processes cannot terminate in HyComp due to out of memory (requiring more than 3GB). However, QBMC can solve it using less than 500 MB. Thus, we can point out that QBMC is superior than HyComp with respect to the memory usage. Moreover, Figures 2 and 3 also indicate that QBMC is able to solve BMC of hybrid automata faster and uses less memory than dReach. Due to state-space (and formula) explosion, the reduction of memory consumption is one of the major challenges to address. Since QBMC requires a smaller amount of memory usage than other quantifier-free BMC approaches, it may be effective in solving BMC of large scale problems.

TABLE II
LYNCH-SHAVIT MUTUAL EXCLUSION PROTOCOL PERFORMANCE.

Tools	L	k ≤ 4		k ≤ 8		k ≤ 16	
		Time (sec)	Mem (MB)	Time (sec)	Mem (MB)	Time (sec)	Mem (MB)
QBMC	9^2	3.7	52.2	5.1	52.3	25.9	52.7
	9^3	15.5	65.6	31.3	87.5	1091.5	144.5
	9^4	256.1	702.8	1062.1	708.9	43578	1196.2
HyComp	9^2	0.8	121.9	1.33	132.8	9.5	170.5
	9^3	2.7	307.9	12.81	380.8	192.8	771.4
	9^4	63.9	2655.4	N/A	M/O	N/A	M/O

We also evaluate QBMC with the Lynch-Shavit mutual exclusion protocol. The Lynch-Shavit protocol is a modified version of Fischer protocol where the mutual exclusion property is time-independent. Each process of Lynch-Shavit protocol has 9 states (locations), then the Lynch-Shavit protocol with 4 processes includes 6561 discrete locations. The performance analyzing the Lynch-Shavit protocol using QBMC and Hycomp are shown in Table II, respectively. M/O presents that the peak memory usage is higher than 3GB, and N/A denotes that the information of running times is not detected due to M/O. The set of bad states of Lynch-Shavit protocol is defined similar to that of Fischer, where two processes may be in the critical section. Again, we can see the trade off between using QBMC or using HyComp. HyComp is faster than QBMC, but requires a higher memory usage. Therefore, the BMC of Lynch-Shavit protocol with 4 processes can be solved by QBMC up to $k = 16$, but cannot be solved in HyComp up to $k = 8$ due to M/O.

V. CONCLUSION AND FUTURE WORK

In this paper, we present a new SMT-based technique that encodes, in a quantified form, the BMC problem for rectangular hybrid automata (RHA), which also subsumes this encoding for timed automata. The preliminary results for the Fischer mutual exclusion protocol and Lynch-Shavit protocol indicate the capability of our method to solve the BMC problem for hybrid systems including more than a thousand locations. We compare these experimental results to those of quantifier-free BMC approaches, such as in the dReach tool that uses the dReal SMT solver, and the HyComp tool built on top of nuXmv that uses the MathSAT SMT solver. As solvers for fragments of many-sorted first-order logic such as LRA, NRA, etc. continue to improve, QBMC encodings such

[2] The preliminary implementation described in this paper, along with all the examples, is available online at: http://www.verivital.com/hyst/cfv2015.zip.

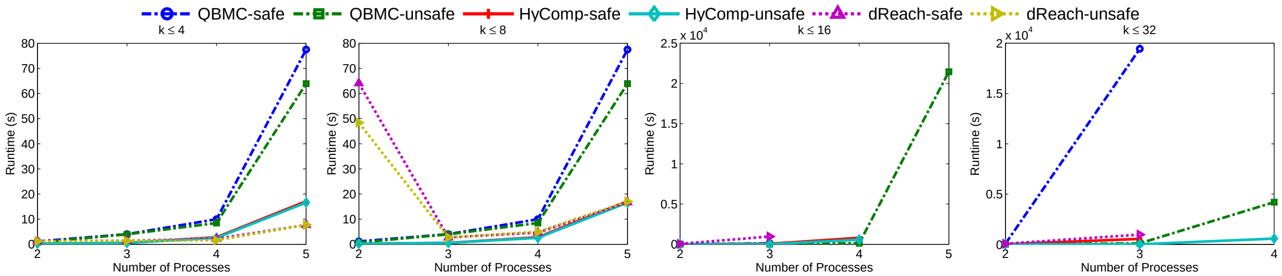


Fig. 2. Runtime comparison of HyComp, dReach and QBMC in solving the BMC of Fischer protocol.

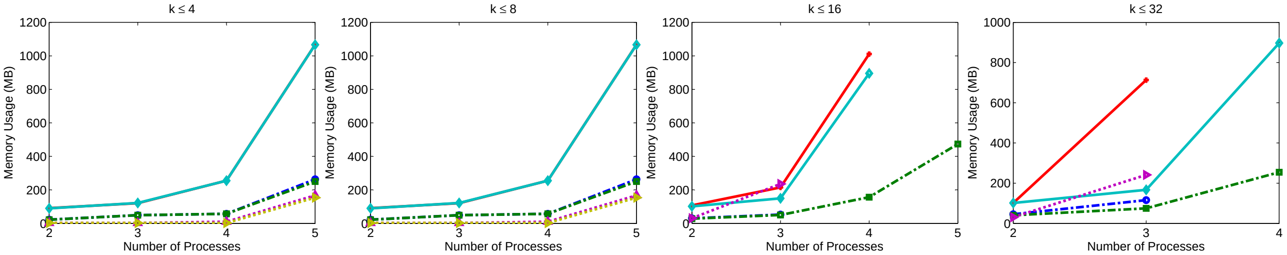


Fig. 3. Memory usage comparison of HyComp, dReach and QBMC in solving the BMC of Fischer protocol.

as the one described in this paper will become more effective, similar to how QBMC for discrete systems has been shown to be effective with QBF encodings [9]. In future work, we will conduct additional experiments and compare the results to other tools and techniques, such as UPPAAL, and also investigate more general classes of hybrid automata, such as those with linear or polynomial differential equations.

Acknowledgments

This material is based on research sponsored by the Air Force Research Laboratory under agreement number FA8750-15-1-0105, the Air Force Office of Scientific Research (AFOSR), in part through the Summer Faculty Fellowship Program (SFFP) and contract FA9550-15-1-0258. This material is based upon work supported by the National Science Foundation (NSF) under Grant Nos. 1464311 and 1527398. The U.S. government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of AFOSR, AFRL, or NSF.

References

[1] A. Eggers, M. Fränzle, and C. Herde, "SAT modulo ODE: A direct SAT approach to hybrid systems," in *Automated Technology for Verification and Analysis*, ser. Lecture Notes in Computer Science, S. Cha, J.-Y. Choi, M. Kim, I. Lee, and M. Viswanathan, Eds. Springer Berlin / Heidelberg, 2008, vol. 5311, pp. 171–185.

[2] T. T. Johnson and S. Mitra, "A small model theorem for rectangular hybrid automata networks," in *Proceedings of the IFIP International Conference on Formal Techniques for Distributed Systems, Joint 14th Formal Methods for Open Object-Based Distributed Systems and 32nd Formal Techniques for Networked and Distributed Systems (FMOODS-FORTE)*, ser. LNCS. Springer, June 2012, vol. 7273.

[3] A. Cimatti, S. Mover, and S. Tonetta, "A quantifier-free smt encoding of non-linear hybrid automata," in *Formal Methods in Computer-Aided Design (FMCAD), 2012*, 2012, pp. 187–195.

[4] S. Gao, S. Kong, and E. Clarke, "Satisfiability modulo ODEs," in *International Conference on Formal Methods in Computer-Aided Design (FMCAD)*, Oct. 2013.

[5] T. T. Johnson and S. Mitra, "Anonymized reachability of rectangular hybrid automata networks," in *Formal Modeling and Analysis of Timed Systems (FORMATS)*, 2014.

[6] A. Cimatti, A. Griggio, S. Mover, and S. Tonetta, "HyComp: An SMT-based model checker for hybrid systems," in *Tools and Algorithms for the Construction and Analysis of Systems*, ser. Lecture Notes in Computer Science, C. Baier and C. Tinelli, Eds. Springer Berlin Heidelberg, 2015, vol. 9035, pp. 52–67.

[7] T. A. Henzinger, "The theory of hybrid automata," in *IEEE Symposium on Logic in Computer Science (LICS)*. Washington, DC, USA: IEEE Computer Society, 1996, p. 278.

[8] T. Jussila and A. Biere, "Compressing bmc encodings with qbf," *Electronic Notes in Theoretical Computer Science*, vol. 174, no. 3, pp. 45–56, 2007.

[9] H. Mangassarian, A. Veneris, and M. Benedetti, "Robust QBF encodings for sequential circuits with applications to verification, debug, and test," *Computers, IEEE Transactions on*, vol. 59, no. 7, pp. 981–994, Jul. 2010.

[10] C. Barrett, A. Stump, and C. Tinelli, "The SMT-LIB standard: Version 2.0," 2010. [Online]. Available: http://smt-lib.org/

[11] L. De Moura and N. Bjørner, "Z3: An efficient SMT solver," in *Proc. of 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, ser. TACAS '08/ETAPS '08. Springer-Verlag, 2008, pp. 337–340.

[12] A. Cimatti, S. Mover, and S. Tonetta, "Smt-based scenario verification for hybrid systems," *Formal Methods in System Design*, vol. 42, no. 1, pp. 46–66, 2013.

[13] J. Bengtsson, K. Larsen, F. Larsson, P. Pettersson, and W. Yi, "UPPAAL: A tool suite for automatic verification of real-time systems," in *Hybrid Systems III*, ser. LNCS, R. Alur, T. Henzinger, and E. Sontag, Eds. Springer, 1996, vol. 1066, pp. 232–243.

[14] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi, "HyTech: A model checker for hybrid systems," *Journal on Software Tools for Technology Transfer*, vol. 1, pp. 110–122, 1997.

[15] X. Chen, E. Abraham, and S. Sankaranarayanan, "Flow*: An analyzer for non-linear hybrid systems," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, N. Sharygina and H. Veith, Eds. Springer Berlin Heidelberg, 2013, vol. 8044, pp. 258–263.

[16] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, "SpaceEx: Scalable verification of hybrid systems," in *Computer Aided Verification (CAV)*, ser. LNCS. Springer, 2011.

[17] S. Gao, J. Avigad, and E. Clarke, "Delta-decidability over the reals," in *Logic in Computer Science (LICS), 2012 27th Annual IEEE Symposium on*, 2012, pp. 305–314.

[18] R. Cavada, A. Cimatti, M. Dorigatti, A. Griggio, A. Mariotti, A. Micheli, S. Mover, M. Roveri, and S. Tonetta, "The nuxmv symbolic model checker," in *Computer Aided Verification*, ser. Lecture Notes in Computer Science, A. Biere and R. Bloem, Eds. Springer International Publishing, 2014, vol. 8559, pp. 334–342.

[19] C. Miller, K. Gitina, and B. Becker, “Bounded model checking of incomplete real-time systems using quantified smt formulas,” in *Microprocessor Test and Verification (MTV), 2011 12th International Workshop on*, Dec. 2011, pp. 22–27.

[20] T. T. Johnson and S. Mitra, “Invariant synthesis for verification of parameterized cyber-physical systems with applications to aerospace systems,” in *Proceedings of the AIAA Infotech at Aerospace Conference (AIAA Infotech 2013)*, Boston, MA, Aug. 2013.

[21] S. Bak, S. Bogomolov, and T. T. Johnson, “HyST: A source transformation and translation tool for hybrid automaton models,” in *Proc. of the 18th Intl. Conf. on Hybrid Systems: Computation and Control (HSCC)*. ACM, 2015.

[22] A. Cimatti, A. Griggio, S. Mover, and S. Tonetta, “Hycomp: An smt-based model checker for hybrid systems,” in *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2015, pp. 52–67.

[23] S. Gao, S. Kong, and E. M. Clarke, “dreal: An smt solver for nonlinear theories over the reals,” in *Automated Deduction–CADE-24*. Springer, 2013, pp. 208–214.

Appendix

A. Appendix: Additional Experimental Results

In this appendix, we describe additional experimental results of the BMC of the Fischer mutual exclusion protocol using QBMC, HyComp and dReach. Figures 4 and 5 show, respectively, the runtime and memory usage comparison among HyComp, dReach and QBMC for BMC of Fischer protocol. Vertical axises are runtime in seconds and memory usage in megabytes, respectively, and horizontal axises are number of steps, k. The details of running times and memory usages of BMC for the Fischer protocol using these tools are also shown in Table III, where FS, FU denote the safe and unsafe versions of Fischer protocol, respectively, and the number following the hyphen (-) describes a number of processes for each version. In FS, a state where the set of bad states ϕ is satisfied is not reachable, while in FU, a state where ϕ is satisfied is reachable. For instance, FS-2, FU-2 are the safe and unsafe versions of the Fischer protocol with 2 processes, respectively.

Table III shows that the BMC of Fischer protocol with 64 discrete locations can be checked completely up to $k = 32$. Note that T/O means the computation time out ($\geq$ 24 hours), M/O presents that the peak memory usage is higher than 3GB, and N/A denotes that the information of times or memory usages are not detected due to M/O or T/O, respectively. The results of the BMC for unsafe versions of Fischer protocol indicate that QBMC is effective for bug detection. However, as k increases, the higher running time and the greater memory usage are required for the quantified encoding of BMC due to the increasing number of all possible paths from an initial state in the set of initial states to a bad state that does not satisfy the set of safety specifications.

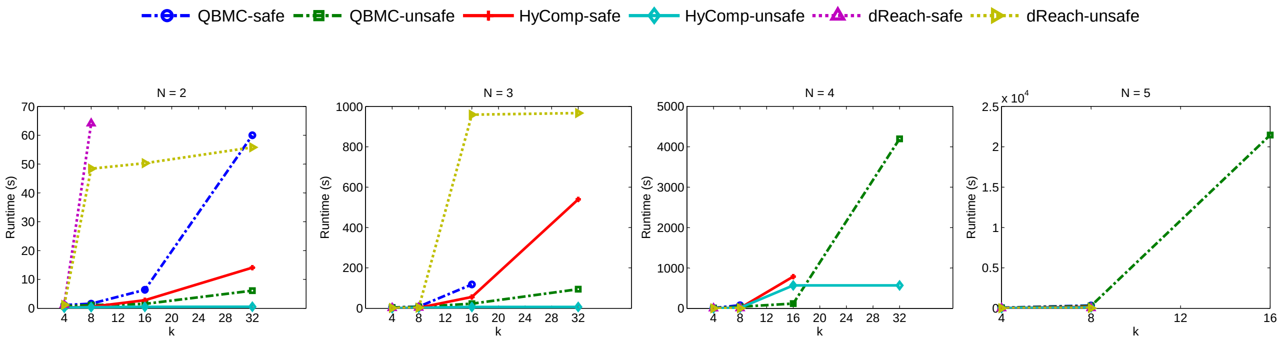


Fig. 4. Runtime comparison of HyComp, dReach and QBMC in solving the BMC of Fischer protocol.

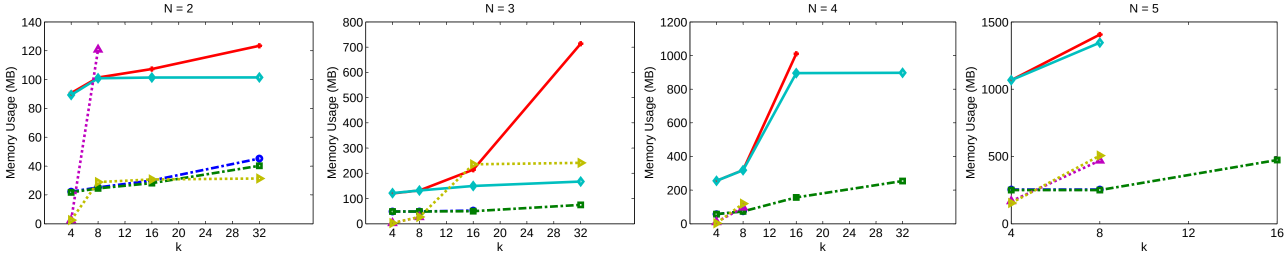


Fig. 5. Memory usage comparison of HyComp, dReach and QBMC in solving the BMC of Fischer protocol.

TABLE III
THE PERFORMANCE OF THE BMC OF FISCHER MUTUAL EXCLUSION PROTOCOL USING QBMC, HYCOMP, AND DREACH.

Tools	Example	L	$k \leq 4$		$k \leq 8$		$k \leq 16$		$k \leq 32$	
			Time (sec)	Mem (MB)	Time (sec)	Mem (MB)	Time (sec)	Mem (MB)	Time (sec)	Mem (MB)
QBMC	FS-2	4^2	1.11	22.3	1.6	25.2	6.4	30	60	45.2
	FU-2	4^2	0.7	21.73	1.1	24.7	1.52	28.2	6.1	40.2
	FS-3	4^3	4.02	48.7	8.3	48.7	117.8	52.4	19452	115.6
	FU-3	4^3	3.97	48.7	6.9	48.7	22.7	49.7	94.3	74.6
	FS-4	4^4	9.97	56.9	76.1	74.1	T/O	N/A	T/O	N/A
	FU-4	4^4	8.44	57	40.1	73.2	119.1	156.2	4197.1	254.1
	FS-5	4^5	77.51	254.3	344.4	254.4	T/O	N/A	T/O	N/A
	FU-5	4^5	63.93	249.9	288.8	249.9	21456	473.8	T/O	N/A
HyComp	FS-2	4^2	0.2	22.3	0.5	101.4	2.8	107.3	14.1	123.4
	FU-2	4^2	0.2	21.7	0.4	100.9	0.5	101.4	0.53	101.5
	FS-3	4^3	0.51	120.2	2.2	131.8	55.8	214.4	539.7	713.4
	FU-3	4^3	0.51	121.5	2.1	131.8	6.7	149.6	6.5	167.1
	FS-4	4^4	2.78	255	9.9	319.1	788	1010.4	T/O	M/O
	FU-4	4^4	2.53	255.2	13.3	318.2	569.4	895.4	568.4	897.1
	FS-5	4^5	17.13	1067	172.4	1405.9	N/A	M/O	N/A	M/O
	FU-5	4^5	16.6	1066.7	109.1	1345.4	N/A	M/O	N/A	M/O
dReach	FS-2	4^2	1.2	2.5	64.1	120.8	T/O	M/O	T/O	M/O
	FU-2	4^2	1.2	2.5	48.4	28.9	50.3	30.7	55.8	31.4
	FS-3	4^3	1.4	2.5	2.7	26.4	T/O	M/O	T/O	M/O
	FU-3	4^3	1.3	2.5	2.7	26.8	959.3	235.3	966.8	241.2
	FS-4	4^4	2.1	9.8	4.63	96.7	T/O	M/O	T/O	M/O
	FU-4	4^4	1.6	2.5	4.93	119.8	T/O	M/O	T/O	M/O
	FS-5	4^5	7.7	167.2	16.69	469.6	T/O	M/O	T/O	M/O
	FU-5	4^5	7.7	153.9	17	506.5	T/O	M/O	T/O	M/O

Runtime Verification for Hybrid Analysis Tools

Luan Viet Nguyen[1], Christian Schilling[2], Sergiy Bogomolov[3], and Taylor T. Johnson[1]

[1] University of Texas at Arlington, USA
[2] Albert-Ludwigs-Universität Freiburg, Germany
[3] IST Austria, Austria

Abstract. In this paper, we present the first steps toward a runtime verification framework for monitoring hybrid and cyber-physical systems (CPS) development tools based on randomized differential testing. The development tools include hybrid systems reachability analysis tools, model-based development environments like Simulink/Stateflow (SLSF), etc. First, hybrid automaton models are randomly generated. Next, these hybrid automaton models are translated to a number of different tools (currently, SpaceEx, dReach, Flow*, HyCreate, and the MathWorks' Simulink/Stateflow) using the HyST source transformation and translation tool. Then, the hybrid automaton models are executed in the different tools and their outputs are parsed. The final step is the differential comparison: the outputs of the different tools are compared. If the results do not agree (in the sense that an analysis or verification result from one tool does not match that of another tool, ignoring timeouts, etc.), a candidate bug is flagged and the model is saved for future analysis by the user. The process then repeats and the monitoring continues until the user terminates the process. We present preliminary results that have been useful in identifying a few bugs in the analysis methods of different development tools, and in an earlier version of HyST.

1 Introduction

Runtime verification is an approach to ensure the correctness and reliability of a system during its execution. It can check and analyze executions of a system under scrutiny that violate or satisfy a given correctness property by using a decision procedure called a monitor. A monitor can also be considered as a device that can read finite traces and output a truth value derived from a truth domain [3]. Runtime verification can be used broadly for many purposes such as debugging, testing, verification, validation, fault protection, and online system repair. In this paper, we describe a preliminary work toward a randomized differential testing framework [5] that may be used as a runtime monitor for various components (from parsers to analysis algorithms) in hybrid and CPS analysis tools such as SpaceEx, dReach, Flow*, HyCreate and the Mathworks' Simulink/Stateflow (SLSF). A test subject is the hybrid automaton randomly generated in the input format for SpaceEx using a prototype tool called HyRG [4][4], which is then translated to other formats including dReach,

[4] The tool and examples are available online: http://www.verivital.com/hyrg/

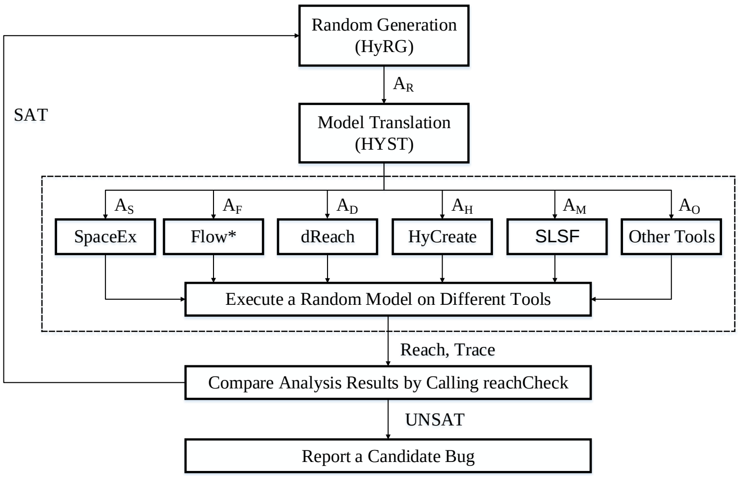


Fig. 1: Overview of monitoring framework for hybrid systems analysis tools with randomized differential testing.

Flow*, HyCreate and SLSF using the HyST model transformation tool [1]. Our contributions include (a) the first steps toward a randomized differential testing framework to monitor CPS development and verification tools, and (b) identifying some bugs in existing tools, including a semantic difference between SpaceEx and SLSF that we did not know about and some soundness bugs in the verification tools that have been corrected by the tool authors [1].

2 Monitoring with Randomized Differential Testing

We first describe how the hybrid systems are randomly generated in HyRG so they have diverse continuous and discrete behaviors. We then analyze these examples with different hybrid systems development and verification tools, and then compare their outputs to identify possible bugs in the tools. Figure 1 shows the overview of our framework for randomized differential testing to monitor hybrid systems development tools. First, a hybrid automaton A_R is randomly generated by HyRG, then A_R is translated using HyST to equivalent automata in different tools' formats, denoted A_S, A_F, A_D, A_H, A_M, and A_O. Next, the automata can be analyzed using the different tools, such as SpaceEx, Flow*, dReach, and HyCreate, or simulated in SLSF. Then we compare all analysis results by using a function reachCheck shown in Figure 2.

The reachCheck function has three inputs: Reach, Trace, and β, where β is the reachability analysis and simulation time bound. Reach is a list of sets of time-bounded reachable states computed by different tools (e.g., the output of SpaceEx, Flow*, etc.). Each set of reachable states, $\mathcal{R}(t)$, is the set of states that may be visited by following the model's trajectories and transitions, for any time $t \in [0, \beta]$. That is, for a given time t, $\mathcal{R}(t)$ is the set of states reachable at time t (sometimes referred to as a time-slice). The input Trace is a set of all simulation traces produced by SLSF up to a maximum simulation time β.

```
function reachCheck(Reach, Trace, $\beta$)
  foreach set of reachable states $\mathcal{R}_i$ in Reach
    foreach set of reachable states $\mathcal{R}_j$ in Reach
      if $i \neq j$ and $\forall t \in [0, \beta]$ $\mathcal{R}_i(t) \wedge \mathcal{R}_j(t)$ is UNSAT then return UNSAT
    foreach execution trace $\mathcal{T}_k$ in Trace
      if $\forall t \in [0, \beta]$ $\mathcal{T}_k(t) \wedge \mathcal{R}_i(t)$ is UNSAT then return UNSAT
  return SAT
```

Fig. 2: reachCheck checks whether the set of reachable states and traces computed by different tools overlap (have non-empty intersection) at every time instant.

The reachCheck function can check whether the reachable states or simulation traces computed by different tools at each time have non-empty intersections. Although all of the reachable states and simulation traces are described in different formats such as support functions, Satisfiability Modulo Theories (SMT) formulas, convex sets, etc., there still exists an equivalence among them. For example, reachable sets computed by SpaceEx's LGG algorithm are a representation of convex sets (support functions), but these could be compared to the Taylor models of Flow*. If the reachable sets computed by different tools have a non-empty intersection (pairwise over all the tools), then reachCheck will return SAT, and the monitoring continues by generating a different random model. Otherwise, there is possibly a bug in the HyST translation or the verification tools. For the simulation traces, if some portions of a trace are not contained in any of the reachable states, reachCheck will return UNSAT and there is again possibly a bug in HyST, the verification tools, or SLSF. Obviously all these tools have numerous parameters, so numerical issues, accuracies, etc. must be taken into account by the user to determine whether a candidate bug is real.

Next, we define the structure of a hybrid automaton [2] and then summarize the random generation framework.

Definition 1. *A hybrid automaton $\mathcal{H}$ is a tuple, $\mathcal{H} \triangleq \langle \texttt{Loc}, \texttt{Var}, \texttt{Flow}, \texttt{Inv}, \texttt{Trans}, \texttt{Init} \rangle$, consisting of following components: (a)* `Loc`*: a finite set of discrete locations. (b)* `Var`*: a finite set of n continuous, real-valued variables, where $\forall x \in \texttt{Var}$, $v(x) \in \mathcal{R}$ and $v(x)$ is a valuation—a function mapping x to a point in its type—here, $\mathcal{R}$; and $\mathcal{Q} \triangleq \texttt{Loc} \times \mathcal{R}^n$ is the state space. (c)* `Inv`*: a finite set of invariants for each discrete location, $\forall l \in \texttt{Loc}$, $\texttt{Inv}(l) \subseteq \mathcal{R}^n$. (d)* `Flow`*: a finite set of derivatives for each continuous variable $x \in \texttt{Var}$, and $\texttt{Flow}(l, x) \subseteq \mathcal{R}^n$ that describes the continuous dynamics in each location $l \in \texttt{Loc}$. (e)* `Trans`*: a finite set of transitions between locations; each transition is a tuple $\tau = \langle \texttt{src}, \texttt{dst}, \texttt{Grd}, \texttt{Rst} \rangle$, which can be taken from source location* `src` *to destination location* `dst` *when a guard condition* `Grd` *is satisfied, and a state is updated by an update map* `Rst`*. (f)* `Init`*: an initial condition, $\texttt{Init} \subseteq \mathcal{Q}$.*

We denote a hybrid automaton that has been randomly generated by A_R. We randomly generate each syntactic component of the automaton A_R. Rather than picking only random matrices and vectors for the affine functions used in flows, guards, invariants, assignments, etc., we instead partition these affine functions into classes. While we assume affine functions making up the automaton, the general method may be extended to nonlinear functions. We highlight that *all* structural components of the automaton are selected randomly (i.e., the tran-

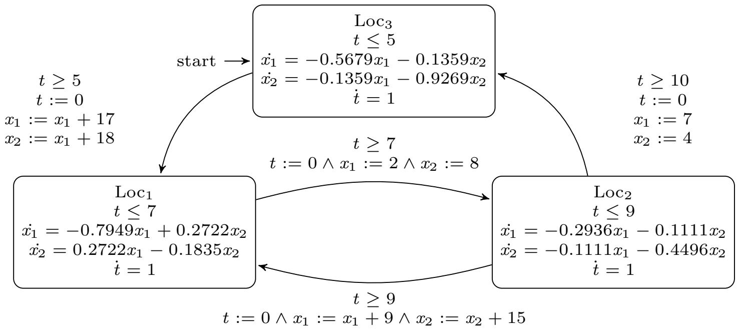

Loc$_3$
$t \leq 5$
$\dot{x_1} = -0.5679x_1 - 0.1359x_2$
$\dot{x_2} = -0.1359x_1 - 0.9269x_2$
$\dot{t} = 1$
start
$t \geq 5$
$t := 0$
$x_1 := x_1 + 17$
$x_2 := x_1 + 18$
$t \geq 10$
$t := 0$
$x_1 := 7$
$x_2 := 4$
$t \geq 7$
$t := 0 \wedge x_1 := 2 \wedge x_2 := 8$
Loc$_1$
$t \leq 7$
$\dot{x_1} = -0.7949x_1 + 0.2722x_2$
$\dot{x_2} = 0.2722x_1 - 0.1835x_2$
$\dot{t} = 1$
Loc$_2$
$t \leq 9$
$\dot{x_1} = -0.2936x_1 - 0.1111x_2$
$\dot{x_2} = -0.1111x_1 - 0.4496x_2$
$\dot{t} = 1$
$t \geq 9$
$t := 0 \wedge x_1 := x_1 + 9 \wedge x_2 := x_2 + 15$

Fig. 3: An example hybrid automaton $\mathsf{A_R}$ with time-dependent switching that was randomly generated using HyRG.

sitions and continuous dynamics), and are not simply parameters. For brevity, we do not describe in detail the random generation of all structural components here, but refer to our other preliminary results [4].

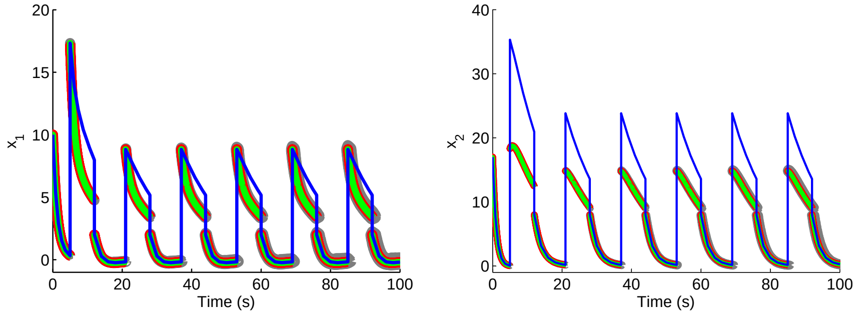
3 Preliminary Experimental Results

We evaluate our preliminary[5] monitoring framework in several scenarios to compare differences among several hybrid systems verification tools including SpaceEx, dReach, and Flow*, as well as SLSF simulation. Consider a randomly generated hybrid automaton $\mathsf{A_R}$ shown in Figure 3. The initial state of $\mathsf{A_R}$ is Loc$_3$, and the randomly generated initial values of its variables are respectively $x_1 = 10$, $x_2 = 17$, and $t = 0$. Note that $\mathsf{A_R}$ is nondeterministic. The results of simulations and reachability analysis on $\mathsf{A_R}$ are shown in Figure 4. The reachable states restricted to x_1 and x_2 computed by Flow* as well as the STC and LGG algorithms in SpaceEx do not contain a simulation trace for a supposedly equivalent SLSF model created using HyST when $\mathsf{A_R}$ takes a transition. In this case, the reachCheck function in Figure 2 will return UNSAT. This happens because of semantic differences in resets among Flow*, SpaceEx, and SLSF. In SLSF, the variables x_1 and x_2 are updated sequentially, so that x_1 will first be updated to a new value, and then x_2 will be updated using the new (already updated) value of x_1. However, these variables are updated concurrently in Flow* and SpaceEx [2], so x_2 will be updated by using the previous value of x_1. Based on this, we fixed this translation error in HyST.

4 Conclusion and Future Work

In this paper, we describe our preliminary results toward building a randomized differential testing framework to monitor hybrid and CPS development tools like SLSF and verification tools like SpaceEx, dReach, Flow*, etc. Our preliminary results include identifying semantic mismatches between tools automatically that have been integrated into subsequent versions of HyST. Additionally, we have found a couple bugs in some of the verification tools that have been corrected by

[5] Some of the steps are currently manual, particularly the parsing of reachable states and comparison thereof, but the generation with HyRG and translation with HyST is fully automatic.



(a) (b)

Fig. 4: SLSF simulation (blue), reachable states computed by Flow* (green), SpaceEx's STC algorithm (red), and SpaceEx's LGG algorithm (gray) for A_R showing x_1 and x_2 versus time, respectively. The SLSF simulation traces and the reachable states computed by Flow*, SpaceEx's LGG and STC algorithms do not line up (i.e., have an empty intersection) at some points in time (so reachCheck returns UNSAT) due to a semantic difference.

the tool authors. Based on our promising preliminary results, we plan to fully automate every step of the framework in the future.

Acknowledgments

This material is based on research sponsored by Air Force Research Laboratory under agreement number FA8750-15-1-0105. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory or the U.S. Government. This work was also partly supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Center Automatic Verification and Analysis of Complex Systems (SFB/TR 14 AVACS, http://www.avacs.org/), by the European Research Council (ERC) under grant 267989 (QUAREM) and by the Austrian Science Fund (FWF) under grants S11402-N23 (RiSE) and Z211-N23 (Wittgenstein Award).

References

1. Bak, S., Bogomolov, S., Johnson, T.T.: HyST: A source transformation and translation tool for hybrid automaton models. In: Proc. of the 18th Intl. Conf. on Hybrid Systems: Computation and Control (HSCC). ACM (2015)
2. Frehse, G., Le Guernic, C., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., Maler, O.: SpaceEx: Scalable verification of hybrid systems. In: Computer Aided Verification (CAV). LNCS, Springer (2011)
3. Leucker, M., Schallhart, C.: A brief account of runtime verification. Journal of Logic and Algebraic Programming 78(5), 293–303 (May 2009)
4. Nguyen, L.V., Schilling, C., Bogomolov, S., Johnson, T.T.: Poster: Hyrg: A random generation tool for affine hybrid automata. In: 18th International Conference on Hybrid Systems: Computation and Control (HSCC 2015) (2015)
5. Yang, X., Chen, Y., Eide, E., Regehr, J.: Finding and understanding bugs in c compilers. In: Proceedings of the 32Nd ACM SIGPLAN Conference on Programming Language Design and Implementation. pp. 283–294. PLDI '11, ACM, New York, NY, USA (2011)

Abnormal Data Classification Using Time-Frequenc Temporal Logic

Luan Viet Nguyen
University of Texas at Arlington, USA

James Kapinski
Toyota Technical Center

Xiaoqing Jin
Toyota Technical Center

Jyotirmoy V. Deshmukh
Toyota Technical Center

Ken Butts
Toyota Technical Center

Taylor T. Johnson
Vanderbilt University, USA

ABSTRACT

We present a technique to investigate abnormal behaviors of signals in both time and frequency domains using an extension of time-frequency logic that uses the continuous wavelet transform. Abnormal signal behaviors such as unexpected oscillations, called *hunting* behavior, can be challenging to capture in the time domain; however, these behaviors can be naturally captured in the time-frequency domain. We introduce the concept of parametric time-frequency logic and propose a parameter synthesis approach that can be used to classify hunting behavior. We perform a comparative analysis between the proposed algorithm, an approach based on support vector machines using linear classification, and a method that infers a signal temporal logic formula as a data classifier. We present experimental results based on data from a hydrogen fuel cell vehicle application and electrocardiogram data extracted from the MIT-BIH Arrhythmia Database.

1. INTRODUCTION

For the last decade, signal temporal logic (STL) [11] has been successfully extended and applied in many domains such as exploring requirements for closed-loop control systems [8], identifying oscillatory behaviors of biology systems [5], and formalizing and recognizing music melodies [7]. Recently, Kapinski et al. introduced a new signal library template for constructing formal requirements of automotive control applications using STL [10]. These requirements involve various control signal behaviors such as settling time, overshoot, and steady state errors. Although most of such control signal behaviors can be characterized in the time domain, some abnormal signal behaviors such as *hunting* (undesirable oscillations) or *spikes* (abrupt, momentary jumps in signal values) are challenging to capture without frequency information. In most practical control systems, hunting behaviors are considered undesirable, or at least not ideal, and care is taken to minimize or eliminate the behavior. In signal processing, hunting behavior can manifest around sharp transitions, as a result of compression artifacts; this occurs, for example, in image processing, resulting in ghostly bands near edges, or in audio compression, resulting in forward echo problems. In circuit design, a hunting behavior can be the unwanted oscillation of an output current or voltage, which may cause a significant rise in power consumption, temperature, electromagnetic radiation, or settling time [9]. Although some hunting behaviors can be defined loosely as an oscillation around a given average and can be well captured using STL, some modulated hunting signals are challenging to detect using only time domain information [10]. Because hunting signals relate to oscillatory properties, it is appropriate to investigate them using time-frequency analysis.

The first attempt to introduce a specification formalism for both time and frequency properties of a signal, called time-frequency logic (TFL), was proposed by Donzé and his collaborators [7]. There, a signal is preprocessed using a Short-Time Fourier Transform (STFT) [4] to generate a spectral signal that represents the evolution of the STFT coefficients at some particular frequency over time. The time-frequency predicates and arithmetic expressions constructed from this spectral signal are added into an STL formula to yield a TFL formula. TFL was originally applied to music, though it can be easily extended to other application domains. A key limitation of the approach using the STFT is the inherent trade-off required between resolution in the time domain and resolution in the frequency domain; it is difficult or impossible to obtain satisfactory resolution in both time and frequency using the STFT for the analysis. Such limitations can be overcome using the continuous wavelet transform (CWT).

In the following, we extend the notion of TFL by using the CWT to specify and check time-frequency properties of signals. We introduce the concept of parametric time-frequency logic (PTFL) and use it to perform parameter synthesis for the purpose of classifying hunting behavior. Previous efforts have focused on data classification of time-series signals using STL [2, 3, 8], but identifying some abnormal behaviors such as hunting requires both time and frequency information [10]. Moreover, existing classification methods require an extensive amount of data, and the inferred classifier is often difficult for engineers to interpret. In contrast, our proposed method using PTFL can efficiently classify abnormal behaviors with an interpretable data classifier and requires less data than existing techniques. We note that although the below presentation is focused on one behavior type, it is straightforward to extend the work to detect other abnormal behaviors such as noise, spikes, or

HSCC '17, April 18–20, 2017, Pittsburgh, PA, USA.
© 2017 ACM. ISBN 978-1-4503-4590-3/17/04…$15.00.
DOI: http://dx.doi.org/10.1145/3049797.3049809

other anomalous behavior, in the time-frequency domain. We evaluate the proposed algorithm by comparing the performance against two existing classification techniques: a traditional machine learning technique using a support vector machine with a linear kernel, and a method that infers STL formulae as data classifiers [3]. To perform the evaluation, we use data sets from two different domains, the automotive and medical domains.

2. TIME-FREQUENCY LOGIC USING CWT

Although many control system behaviors can be naturally characterized in the time domain, there are some signal behaviors, such as hunting and spikes, that are challenging to capture without frequency information. This is especially true for non-stationary signals whose frequency components vary over time; for this class of signals, it is essential to analyze the signal properties in the time-frequency domain. STFT is a popular transformation that has been widely used in time-frequency analysis [4]. Using STFT to perform time-frequency analysis, a signal is partitioned into small segments (each segment is assumed to be stationary) whose lengths are equal to the width of a chosen window function. The window function is used to modulate the signal to emphasize the time instant associated with each segment. Unfortunately, the STSF provides a fixed time-frequency resolution so that it is not effective for signals that need to be analyzed with different time-frequency resolutions [14]. Moreover, it is difficult to choose a proper window function with an appropriate size that not only provides both desirable time and frequency resolutions but also does not violate the stationarity condition [14]. To overcome the limitation of the STFT, we use the CWT to analyze a signal in the time-frequency domain.

2.1 Continuous Wavelet Transform

The CWT of a signal $x(t)$ is formally defined as follows:

$$Wf(\zeta,\tau) = \int_{-\infty}^{+\infty} x(t)\psi^*_{\zeta,\tau}(t), \qquad (1)$$

where $\psi^*_{\zeta,\tau}(t)$ is the complex conjugation of a basic wavelet function $\psi_{\zeta,\tau}(t)$ which is derived from a mother-wavelet function $\psi(t)$. This function has zero average in the time domain, i.e. $\int_{-\infty}^{+\infty} \psi(t)dt = 0$. Furthermore, a basic wavelet function $\psi_{\zeta,\tau}(t)$ can be written as:

$$\psi_{\zeta,\tau}(t) = \frac{1}{\sqrt{\zeta}}\psi\left(\frac{t-\tau}{\zeta}\right), \qquad (2)$$

where $\zeta \in \mathbb{R}_{>0}$ is a scale parameter representing the width of the basic wavelet function, $\tau \in \mathbb{R}$ is a translation factor representing the location of the basic wavelet function, and $\frac{1}{\sqrt{\zeta}}$ is the energy normalization across different scales. Thus, the CWT maps an original signal to a function of ζ and τ that provides both time and frequency information. Note that the scale factor is inversely proportional to the frequency of a signal [14]. The CWT in Equation 1 measures the similarity between a basic wavelet function and a signal. Indeed, if a signal $x(t)$ has a frequency component f corresponding to a particular scale ζ of a wavelet function $\psi_{\zeta,\tau}(t)$, then the portion of $x(t)$ at some particular time interval where f exists will be similar to $\psi_{\zeta,\tau}(t)$. As a result, the CWT coefficients of $x(t)$ corresponding to f will be relatively large over this time interval. Moreover, the time-frequency energy density of the CWT is equivalent to the square norm of the CWT coefficients:

$$P_W f(\zeta,\tau) = |Wf(\zeta,\tau)|^2. \qquad (3)$$

Time-frequency resolution. In contrast to the STFT, the CWT can either dilate or compress the window size of the wavelet function, and translate it along the time axis. The Heisenberg box [12] is a range of times and frequencies that indicates the accuracy of a time-frequency transformation. Although the area of the Heisenberg box does not change, the time and frequency resolutions can be varied depending on the value of ζ. As a result, the CWT can analyze all frequency components within a signal by considering appropriate scales of the mother-wavelet function. For instance, the CWT can use the wavelet function with a short duration and low scale for analyzing high frequency components, and vice versa. This advantage of the CWT allows us to efficiently analyze a signal that includes abnormal behaviors such as spikes and hunting.

2.2 Time-Frequency Logic

TFL is an extension of STL that can be used to specify both time and frequency properties of a signal [7]. In TFL, a signal predicate is defined over the signal representing the evolution of the STFT coefficient at a particular frequency over time. Given a pair (f,τ) of frequency and time, the STFT of a signal $x(t)$ is obtained by:

$$S_{f,\tau} = \int_{-\infty}^{+\infty} x(t)\psi_L(t-\tau)e^{-2i\pi ft}dt\,, \qquad (4)$$

where $\psi_L(t)$ is a window function. A *spectral signal* $y(t) = |S_{f,t}|^2$ is the projection of the spectrogram of $x(t)$ on a particular frequency f. Such a signal can be incorporated in TFL formulae to form some interesting time-frequency specifications. We can see that a TFL formula is actually an STL formula in which the signal predicate is defined over $y(t)$ instead of $x(t)$. TFL has been used to formalize and recognize music melodies, where time-frequency requirements are simply specified as $\varphi \triangleq |S_{f_p,t}|^2 > \theta$, where f_p is the pitch frequency and θ is the STFT coefficient threshold [7]; however, the shortcomings of the STFT mentioned previously may reduce the ability of TFL to precisely specify and evaluate time-frequency properties of a signal. We extend TFL to use the CWT to obtain spectral signals from a given time-series signal. In effect, we construct a TFL formula based on the CWT coefficients of the spectral signals instead of the STFT coefficients. Because the CWT can appropriately use various scaling factors, ζ, to analyze all frequency components at different time intervals, it gives us an ability to study signals at flexible time-frequency resolutions.

Although the following presentation focuses on the classification of hunting behaviors, we note that the proposed approach using TFL and CWT can be used to capture other time-frequency specifications as well. For instance, consider the property: "*For some time in the future, the dominant frequency of the signal is ω for 5 time units, and the dominant frequency subsequently rises to twice of this value within 10-time units.*" Here, the *dominant frequency*, $f(t)$, of a signal $x(t)$ is defined as the frequency corresponding to the maximum magnitude frequency component of the signal at time t, as provided by a CWT. Such a time-frequency property can be written as a TFL formula, $\varphi \triangleq \Diamond(\Box_{[0,5]}(f = \omega) \wedge \Diamond_{[5,15]}(f = 2\omega))$. Then, the TFL formula φ can be evaluated as a normal STL formula using Breach[1] [6]. Consider another property such as "*At some time in the future the energy densities of the signal within a particular time inter-*

[1] Breach [6] is a tool that allows evaluation of STL and TFL formulae on signals.

val and a particular frequency bandwidth are always greater than some threshold value θ." This property can be specified as a TFL formula, $\phi \triangleq \Diamond\Box_{[t_1,t_2]}(z(f,t) > \theta)$, where $z(f,t)$ is a spectral signal that captures the minimum value of the CWT coefficients of a signal over some frequency bandwidth $[f_1, f_2]$.

Parametric Time-Frequency Logic. We introduce parametric time-frequency logic (PTFL), which is an extension of TFL where the parameters in TFL template formulae are symbolic parameters. Similar to the concept of parameter signal temporal logic (PSTL) introduced in [1], PTFL allows constants in intervals bounding the temporal operators and constant values in the predicates of PTFL formulae to be replaced with parameters.

The p parameters in a PTFL formula are classified into two sets:

(a) $\Upsilon = \{\tau_1, ..., \tau_{p_t}\}$ is a set of p_t *time* parameters occurring in the time intervals of the temporal operators, and

(b) $\Theta = \{\theta_1, ..., \theta_{p-p_t}\}$ is a set of $p - p_t$ *threshold* parameters occurring in the signal predicates.

For any fixed values of Υ and Θ, a PTFL formula $\varphi(\tau_1, \ldots, \tau_{p_t}, \theta_1, \ldots, \theta_{p-p_t})$ yields a TFL formula corresponding to the fixed values of the parameters. For instance, consider a PTFL formula $\varphi(\tau, \theta) \triangleq \Box_{[0,\tau]}(y(t) > \theta)$, where $y(t)$ is a spectral signal, τ and θ are time and threshold parameters, respectively. The formula $\varphi(5, 10)$ is defined as the TFL formula $\Box_{[0,5]}(y(t) > 10)$.

3. HUNTING CLASSIFICATION

In this section, we will describe three different approaches using PTFL and TFL to efficiently classify hunting behaviors in signals. Informally, a hunting behavior is an undesirable oscillation appearing within a signal over some time interval.

3.1 Parameter Synthesis Approach

We now propose a method to classify hunting behavior based on mining parameters of the following PTFL formula:

$$\varphi_h \triangleq \bigwedge_{i=1}^{m} \Diamond_{[0,\tau_i]}(Wf_i(t) > \theta_i). \qquad (5)$$

Intuitively, this formula specifies that "*the energy densities of the given signal at particular frequencies are eventually greater than some threshold value*". Here, $Wf_i(t)$ is a spectral signal over time that captures the energy densities of the CWT of an original time-series signal $x(t)$ at a particular frequency $f_i \in F$. Note that F is a set of frequencies based on the scales of the CWT. Each spectral signal, $Wf_i(t)$, is the row vector of the matrix representing the energy densities of the CWT of $x(t)$; such a matrix is obtained using Equation 1 and Equation 3. Also, $\tau_i \in \Upsilon$ and $\theta_i \in \Theta$ denote a time and threshold parameter corresponding to each spectral signal $Wf_i(t)$. We note that the satisfaction value of the property φ_h monotonically increases in τ_i and decreases in θ_i. Because of monotonicity, we can exponentially reduce the search over the parameter space so that the synthesis procedure is efficient [8]. Figure 1 conceptually illustrates a spectral signal $Wf_i(t)$, and an instance of a hunting behavior that may occur within a signal. We say that a signal $x(t)$ contains hunting behavior if the property φ_h holds. Overall, the hunting classification problem can be written as follows.

- **Given** the following inputs:

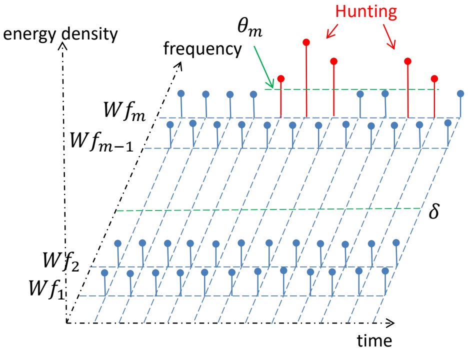


Figure 1: A sketch illustrates the hunting classification problem using time-frequency parameter synthesis. The set of spectral signals Wf_i is acquired from the CWT of an original time-series signal.

  □ a set of labeled traces $\Psi \triangleq \{\Psi_\alpha, \Psi_\beta\}$, where Ψ_α and Ψ_β denote a set of training and testing traces, respectively. Moreover, we the notation $\Psi.B$ and $\Psi.G$ to respectively denote the set of traces with and without hunting behavior. Note that all traces in the training set exhibit hunting behavior, so that $\Psi_\alpha = \Psi_\alpha.B$

  □ a cut-off frequency δ.

  □ sets of parameters Υ, and Θ.

- **Find** values for Υ and Θ, such that:

  □ $x_j(t) \models \varphi_h(\Upsilon, \Theta)$ for all $x_j(t) \in \Psi_\beta.B$.

  □ $x_j(t) \not\models \varphi_h(\Upsilon, \Theta)$ for all $x_j(t) \in \Psi_\beta.G$.

We introduce the cut-off frequency δ to reduce the effort to exhaustively mine parameters over the entire time-frequency domain. It is essential for the control engineers to indicate that hunting behavior only occurs at some high-frequency region above δ.

Classification Algorithm. Next, we propose a heuristic to automatically obtain values for Υ and Θ that can be used to separate the hunting and non-hunting signals. An overview of the heuristic is described in Algorithm 1. The heuristic can be interpreted as follows.

Line 2 initializes a matrix Σ that represents the k m-dimensional spectral signals transformed from k original time-series signals in the training set using the CWT. We iterate over each trace in Ψ_α to construct sets of spectral signals $\{Wf_1(t), ..., Wf_m(t)\}$ using the CWT, and assign them to Σ. Next, we call the function TruncateParam to reduce the effort of exhaustively mining all parameters over the entire time-frequency domain. Here, Σ' represents the k n-dimensional $(n < m)$ matrix of Σ corresponding to the frequency range above δ. Next, we call the function HuntingParamSyn incorporated inside Breach to mine values for Υ and Θ. Then, we test the classifier with a given set of testing traces Ψ_β. The function Classifier checks the satisfaction of φ_h for each trace in Ψ_β, and returns the misclassification rate (MCR) value and the set of misclassified traces Ψ_m. The values of Υ, Θ and the set Ψ_m are then returned for further analysis. Furthermore, we can call EnhancedParam function to strengthen the values Υ and Θ and reduce the MCR value for the purpose of optimizing the classifier formula. Note that in the case studies, we do not use this function to evaluate the performance of the classifier to avoid the bias in our comparative analysis.

3.2 Decision Tree Approach

An approach based on decision trees to classify time series data using STL formulae was implemented in the tool

Algorithm 1 Hunting Classification Using Parameter Synthesis

```
function HuntingClassification($\Psi_\alpha, \Psi_\beta, \delta$)
    $\Sigma \leftarrow 0$
    for each trace $x_j(t) \in \Psi_\alpha,\ j \leq k$
        $\Sigma(j,:,:) \leftarrow Wf_1(t), ..., Wf_m(t) \leftarrow CWT(x_j(t))$
    end for
    $\Sigma' \leftarrow$ TruncateParam($\delta, \Sigma$)
    $\Upsilon, \Theta \leftarrow$ HuntingParamSyn($\Sigma'$)
    $MCR, \Psi_m \leftarrow$ Classifier($\Upsilon, \Theta, \Psi_\beta$)
    return $\Upsilon$, $\Theta$, $\Psi_m$
end function
function EnhancedParam($\Psi_m, \Psi_\alpha, \Psi_\beta, \delta$)
    if $\Psi_m.B \neq \emptyset$ then
        $\Psi'_\alpha \leftarrow \Psi_\alpha \cup \Psi_m.B$
        HuntingClassification($\Psi'_\alpha, \Psi_\beta, \delta$)
    end if
end function
```

DT4STL [3]. That method uses a parameterized procedure to infer STL formulae from labeled data. Given a two-class training data and a set of PSTL templates, a decision tree for classification is recursively built such that each node of a tree is associated with a simple formula, selected from the given PSTL templates. The parameter synthesis is then conducted to find the STL formula that yields the best split for the data at each node. This technique can be used to automatically construct classifiers based on STL formula, but to achieve a low MCR value, the inferred STL formulae may be long and not easily interpretable by engineers. In this section, we apply this approach to classify hunting versus non-hunting signals. Instead of inferring an STL formula, we intend to infer a TFL formula as a data classifier. Thus, we transform original time series data into a collection of time-frequency data (spectral signals).

We assume that control engineers initially designate the frequency threshold separating hunting versus non-hunting behavior. A hunting behavior is specified as any oscillatory behavior occurring at frequencies above some specified cut-off frequency δ. Thus, the time-frequency profile of a hunting signal at some frequency component $f > \delta$ contains larger values for the CWT coefficients compared to those of non-hunting signals. So we define the spectral signal WThcoef based on the CWT coefficients of the signal in a high-frequency region such that:

$$\mathsf{WThcoef}(t) = \max_{\zeta \in [\frac{f_c}{T_s F_{max}}, \frac{f_c}{T_s \delta})} P_W f(\zeta, t), \qquad (6)$$

where f_c is a center frequency associated with the mother-wavelet function, F_{max} is the maximum frequency that appears in the CWT, and T_s is the sampling period. We use such a spectral signal as an input for the DT4STL to infer a simple TFL formula. Note that in this scenario, the inferred TFL formula captures the non-hunting behavior of a signal.

3.3 Support Vector Machine Approach

Next, we present another approach that can solve the problem of hunting classification: linear classification using support vector machines (SVM) [15]. A linear SVM is a set of hyperplanes or decision boundaries that can correctly separate data into two classes. The general form of hyperplanes is $\langle w \cdot x \rangle + b = 0$, where w is a normal to the hyperplane, and $\frac{b}{||w||}$ is the perpendicular distance from the hyperplane to the origin. The sign of the linear discriminant function $f(x) \triangleq \langle w \cdot x \rangle + b$ determines on which side of the decision boundary the test data point is located. The distance from the decision boundary to the closest data point determines the *margin* of the linear classifier. Suppose that we have a set of n labeled training data $(x_i, c_i), ..., (x_n, c_n)$ where $x_i \in \mathbb{R}^d$ and $c_i \in \{1, -1\}$, the constrained optimization problem of linear classification using SVM is written as:

$$\begin{aligned} \underset{w,b}{\text{minimize}} \quad & \frac{1}{2}||w||^2 + C\sum_{i=1}^{n} \zeta_i \\ \text{subject to} \quad & c_i(\langle w \cdot x_i \rangle + b) \geq 1 - \zeta_i,\ i = 1, \ldots, n \\ & \zeta_i \geq 0. \end{aligned} \qquad (7)$$

Here, ζ is a slack variable. If $0 < \zeta \leq 1$, the data point lies somewhere between the margin and the correct side of hyperplane, and the data point is misclassified if $\zeta > 1$. C is a regularization parameter that defines the trade-off between errors of the SVM on training data and margin maximization. A large value of C results in the low possibility of misclassified training data points, because the optimization in Equation 7 will choose a narrow margin hyperplane that correctly separates training data points as much as possible. In contrast, a small value of C will result in a large margin hyperplane, but it may yield a better result in terms of correctly separating testing data points. Due to space limitation, we will not discuss the formal optimization problem solved to obtain the SVM, but refer interested readers to [15]. In this work, instead of applying the linear SVM directly to original time series signals, we need to preprocess them to yield a corresponding set of time-frequency features. For each time-series signal $x(t)$, we collect a real-valued vector $W^{max} \triangleq [Wf_1^{max}, ..., Wf_m^{max}]$ such that each element $Wf_i^{max} \in W^{max}$ is the maximum value of a spectral signal $Wf_i(t)$. Such a vector will be used as a time-frequency feature to design the SVM.

4. CASE STUDIES

In this section, we evaluate the capabilities of three different methods to classify hunting behavior for two case studies. The first case study is based on data from an air compressor motor speed (ACMS) system in a fuel cell (FC) vehicle application. The second case study is based on electrocardiogram (ECG) data. In both examples, we apply the Morlet CWT [12] to the time-series signals.

4.1 ACMS Data

The ACMS system uses a compressor to regulate the air intake of a hydrogen FC vehicle. An FC stack uses a mixture of air and hydrogen to generate electrical power for the vehicle. Accurate control of the compressor which translates to control of the quantities of hydrogen and oxygen (air) is required to achieve good performance and proper operation from the FC stack. Also, the water balance (moisture level) within the stack needs to be carefully regulated, which requires regulation of the air pressure at the inlet of the stack. The task of the ACMS system is to regulate air flow and air pressure delivered to the inlet of the FC stack.

We consider ACMS data from an FC vehicle application. Specifics of the data, such as units and descriptions of the measured quantities are omitted here for proprietary reasons. The ACMS data are partitioned into a collection of traces that are 100 seconds in length and are labeled as either good (the trace does not exhibit hunting behavior) or bad (the trace does exhibit hunting behavior). The ACMS

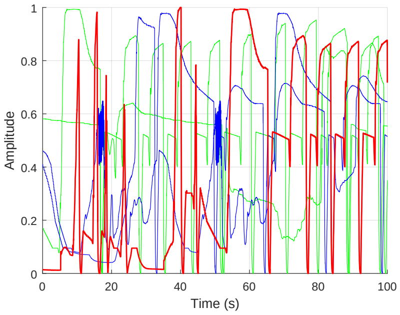


Figure 2: The classified testing data of the ACMS signals using parameter synthesis approach.

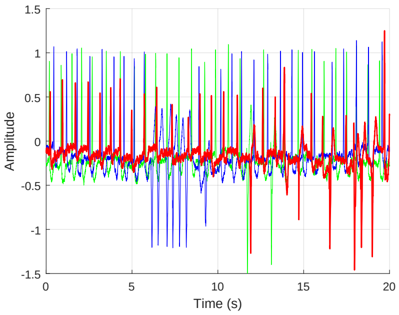


Figure 3: The classified testing data of the ECG signals using parameter synthesis approach.

data has a sampling period of 0.02 seconds. We note that the same training data is used for all of the evaluations, though the parameter synthesis approach only uses the bad traces. In this experiment, we use the training data including 50 total traces, in which 30 traces are labeled as good and the others are labeled as bad. We also use the same testing data including 10 good traces and 10 bad traces for all of the evaluations.

Parameter Synthesis. We now illustrate the performance of the classification heuristic shown in Algorithm 1 to classify hunting behavior for the ACMS signals. Because we do not know the frequency range where a hunting behavior may occur, we exhaustively mine all parameters $\tau_i \in \Upsilon$ and $\theta_i \in \Theta$. We choose the maximum frequency of the CWT as $F_{max} = 25$Hz. Here, the Algorithm 1 will search for the best $\theta_i \in [0, 1]$ and $\tau_i \in [0, 100]$ such that all spectral signals transformed from original time-series traces in the training data satisfy φ_h. We then use Breach with the optimized parameters of φ_h to classify good versus bad traces in the testing set.

Figure 2 shows the experimental results of classifying abnormal ACMS signals, using the function HuntingClassification. In the figure, we only show five representative signals in which good traces correctly classified are shown in green, and bad traces correctly classified are shown in blue. The one good trace that is misclassified is shown in red. The total running time of the classification process is approximately 3 minutes.

Decision Tree Approach. Next, we utilize the DT4STL toolbox to infer TFL formulae that can be used to classify hunting behavior for the ACMS data.

We preprocess the training data to yield the corresponding set of spectral signals WThcoef with $\delta = 15$Hz and $F_{max} = 25$Hz. We then run the DT4STL toolbox with this set of spectral signals using 2-fold cross-validation. As a result, we obtain the two following TFL formulae:

$$\varphi_{h1} \triangleq \square_{[37.4,98.2)}(\mathsf{WThcoef} < 0.0435)$$

$$\varphi_{h2} \triangleq \square_{[1.29,91.3)}(\mathsf{WThcoef} < 0.0394).$$

The procedure takes approximately 75 seconds to infer each formula. Using Breach, we then evaluate those formulae with the set of testing data. The formula φ_{h1} gives us all misclassified traces that are bad traces with the MCR value being equal to 25%. On the other hand, the formula φ_{h2} results in one misclassified trace, which is a bad trace.

SVM Approach. We apply the SVM method to classify normal versus abnormal ACMS data. We first transform all of the traces in the training data into sets of time-frequency features. Next, we run the linear SVM to learn the decision boundaries that separate data as either good or bad. Finally, we predict the testing data from the learned decision boundaries with different values of the SVM classifier margin C.

The MCR of the hunting classification for the ACMS data using SVM is 10% with $C = 10$ and reduces to 5% with $C = 100$. In this case, a larger value of C gives a better result for the classification. Moreover, the classification process takes only 0.393 seconds.

4.2 ECG Data

An electrocardiogram (ECG) test is a noninvasive procedure used to monitor the electrical activities of a heart via a collection of electrodes attached to the patient's skin. A doctor can read an ECG output signal to diagnose abnormal structure or function of the patient's heart. A normal ECG signal includes three signals: (a) the P wave representing the depolarization or contraction of the atrium (b) the QRS complex (the R wave) indicating the ventricular depolarization and (c) the T wave describing the ventricular repolarization. The distance between two consecutive R peaks is considered as a heartbeat. A healthy patient has a resting normal heartbeat (frequency) from 60 to 100 beats per minute (bpm).

In this paper, we focus on classifying the ECG signal that may contain a ventricular tachycardia (VT), a very fast heart rhythm arising in the ventricles that may cause a sudden heart failure. VT is defined as a sequence of three or more ventricular beats with the frequency varying from 110 to 250 bpm. Thus, a VT can be considered as a hunting behavior in an ECG signal. We conduct our classification approaches on the MIT-BIH Arrhythmia ECG Database. These data contain a variety of ECG signals collected from patients 23 to 89 years of age, including patients who experience ventricular arrhythmia [13]. We transform ECG signals 20 seconds in duration (provided at a sampling period of 0.0028 secs.) to spectral signals using the Morlet CWT. Here, the maximum frequency of the CWT is $F_{max} = 4.5$Hz ($\sim$ 270 bpm). For all of the evaluations, we use the same training data including 20 bad traces (the traces do contain a VT) and 40 good traces (the traces do not contain a VT), and the same testing data including 10 good traces and 10 bad traces.

Parameter Synthesis. In this scenario, we only mine the parameters for 20 bad traces in the training dataset. Here, we will search for the best $\theta_i \in [0, 5]$ and $\tau_i \in [0, 20]$. Figure 3 shows the experimental results of using the function HuntingClassification to classify abnormal ECG signals that contain VT. Here, we only show three signals for illustration. The approach results in one (5%) misclassified (red)

	PS	DT4STL	SVM
Interpretation of data classifier	○	△	×
Computation time	×	×	○
Bad behavior localization	○	○	×
Low misclassification rate	△	△	○

Table 1: The comparison between parameter synthesis (PS) using PTFL, DT4STL toolbox using TFL, and linear SVM in classifying abnormal signals, where ○, △, × respectively denote good, ok, bad.

trace, which is a bad trace. The total running time of the classification process is approximately 1 minute.

Decision Tree Approach. Next, we utilize the DT4STL toolbox to classify hunting behavior for the ECG data. We first preprocess the training data to yield the corresponding set of spectral signals WThcoef with $\delta = 1.5$Hz. Then, we run the DT4STL toolbox with this set of spectral signals using 2-fold cross-validation. As a result, we obtain two following TFL formulae:

$$\phi_{h1} \triangleq \Box_{[1.73,17.3)}(\mathsf{WThcoef} < 3.16)$$
$$\phi_{h2} \triangleq \Box_{[2.36,20)}(\mathsf{WThcoef} < 3.21).$$

The procedure takes approximately 105 seconds to infer each formula. We then use Breach to evaluate these formulae with a set of spectral data acquired from the CWT of 10 good traces and 10 bad traces in the testing data. The MCR values of using ϕ_{h1} and ϕ_{h2} to classify these data are both equal to 5% (but misclassified traces are different).

SVM Approach. Finally, we apply the SVM approach to classify hunting in the ECG data. Note that we use the same training and testing data used for the other methods. The hunting classification of the ECG data using an SVM results in a 5% MCR for all values of C (the one misclassified trace is a bad trace), and the classification procedure takes 0.3 seconds.

5. DISCUSSION

In this section, we discuss the trade-offs related to the three classification approaches presented above to classify normal versus abnormal signals. Table 1 shows an aggregate performance evaluation between the approaches in four different categories, including (a) the ability to interpret the structure and parameters used to define the classifier, (b) the computation time, (c) the capacity to localize where bad behavior occurs in a signal, and (d) the ability to correctly classify normal versus abnormal signals. Although the linear SVM can classify abnormal signals much faster and more accurately than the parameter synthesis and the decision tree approaches, the main drawback of this method is that it cannot reveal where the bad behavior occurs within a signal. We found that the decision tree approach can infer specifications that accurately classify data as either good or bad; however, it is not easy to interpret the inferred formula unless the user has some expertise about the input data. If a dataset is not homogeneous (i.e., both normal and abnormal signals are very different from each other), the DT4STL toolbox may infer a complicated formula that cannot be easily interpreted. The parameter synthesis using PTFL and the decision tree approach using TFL have similar performance except the former provides a clearer intuition about the classifier, as the temporal logic formula that results is usually simpler for the PTFL case. Overall, we conclude that a traditional machine learning technique such as the linear SVM is the best choice if the only goal is to classify data as either good or bad, and the most important thing is to select a proper feature on which to base the classification algorithm. Otherwise, if the designer additionally wishes to both understand the meaning of a data classifier and automatically localize where abnormal behaviors occur within a signal, we conclude that the parameter synthesis approach is the best option, as a simple temporal logic formula that defines the classifier results from the analysis.

6. ACKNOWLEDGMENTS

The material presented in this paper is based upon work supported by the National Science Foundation (NSF) under grant numbers CNS 1464311, EPCN 1509804, and SHF 1527398, the Air Force Research Laboratory (AFRL) through contract number FA8750-15-1-0105, and the Air Force Office of Scientific Research (AFOSR) under contract numbers FA9550-15-1-0258 and FA9550-16-1-0246. The U.S. government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of AFRL, AFOSR, or NSF. The authors sincerely appreciate Jared Farnsworth and the Fuel Cell group at Toyota Technical Center for their help in obtaining and understanding the ACMS data.

7. REFERENCES

[1] E. Asarin, A. Donzé, O. Maler, and D. Nickovic. Parametric identification of temporal properties. In *Runtime Verification*, pages 147–160. Springer, 2011.

[2] E. Bartocci, L. Bortolussi, and G. Sanguinetti. Data-driven statistical learning of temporal logic properties. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 23–37. Springer, 2014.

[3] G. Bombara, C.-I. Vasile, F. Penedo, H. Yasuoka, and C. Belta. A decision tree approach to data classification using signal temporal logic. In *Proceedings of the 19th international conference on Hybrid systems: computation and control*. ACM, 2016.

[4] L. Cohen. *Time-frequency analysis*, volume 299. Prentice hall, 1995.

[5] P. Dluhoš, L. Brim, and D. Šafránek. On expressing and monitoring oscillatory dynamics. *arXiv preprint arXiv:1208.3853*, 2012.

[6] A. Donzé. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *Computer Aided Verification*, pages 167–170. Springer, 2010.

[7] A. Donzé, O. Maler, E. Bartocci, D. Nickovic, R. Grosu, and S. Smolka. On temporal logic and signal processing. In *Automated Technology for Verification and Analysis*, pages 92–106. Springer, 2012.

[8] X. Jin, A. Donzé, J. V. Deshmukh, and S. A. Seshia. Mining requirements from closed-loop control models. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(11):1704–1717, 2015.

[9] H. W. Johnson, M. Graham, et al. *High-speed digital design: a handbook of black magic*, volume 1. Prentice Hall Upper Saddle River, NJ, 1993.

[10] J. Kapinski, X. Jin, J. Deshmukh, A. Donze, T. Yamaguchi, H. Ito, T. Kaga, S. Kobuna, and S. Seshia. ST-Lib: A library for specifying and classifying model behaviors. 2016.

[11] O. Maler and D. Nickovic. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pages 152–166. Springer, 2004.

[12] S. Mallat. *A wavelet tour of signal processing*. Academic press, 1999.

[13] G. B. Moody and R. G. Mark. The impact of the mit-bih arrhythmia database. *IEEE Engineering in Medicine and Biology Magazine*, 20(3):45–50, 2001.

[14] R. Polikar. The wavelet tutorial. 1996.

[15] B. Scholkopf and A. J. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond.* MIT press, 2001.

Hyperproperties of Real-Valued Signals

Luan Viet Nguyen
University of Texas at Arlington
luanvnguyen@mavs.uta.edu

James Kapinski
Toyota Motor North America R&D
jim.kapinski@toyota.com

Xiaoqing Jin
Toyota Motor North America R&D
xiaoqing.jin@toyota.com

Jyotirmoy V. Deshmukh
Toyota Motor North America R&D
jyotirmoy.deshmukh@toyota.com

Taylor T. Johnson
Vanderbilt University
taylor.johnson@vanderbilt.edu

ABSTRACT

A *hyperproperty* is a property that requires two or more execution traces to check. This is in contrast to properties expressed using temporal logics such as LTL, MTL and STL, which can be checked over individual traces. Hyperproperties are important as they are used to specify critical system performance objectives, such as those related to security, stochastic (or average) performance, and relationships between behaviors. We present the first study of hyperproperties of cyber-physical systems (CPSs). We introduce a new formalism for specifying a class of hyperproperties defined over real-valued signals, called HyperSTL. The proposed logic extends signal temporal logic (STL) by adding existential and universal trace quantifiers into STL's syntax to relate multiple execution traces. Several instances of hyperproperties of CPSs including stability, security, and safety are studied and expressed in terms of HyperSTL formulae. Furthermore, we propose a testing technique that allows us to check or falsify hyperproperties of CPS models. We present a discussion on the feasibility of falsifying or verifying various classes of hyperproperties for CPSs. We extend the quantitative semantics of STL to HyperSTL and show its utility in formulating algorithms for falsification of HyperSTL specifications. We demonstrate how we can specify and falsify HyperSTL properties for two case studies involving automotive control systems.

ACM Reference format:
Luan Viet Nguyen, James Kapinski, Xiaoqing Jin, Jyotirmoy V. Deshmukh, and Taylor T. Johnson. 2017. Hyperproperties of Real-Valued Signals. In *Proceedings of MEMOCODE '17, Vienna, Austria, September 29-October 2, 2017,* 10 pages.
DOI: 10.1145/3127041.3127058

1 INTRODUCTION

Hyperproperties were first proposed by Clarkson and Schneider to characterize properties of security policies that cannot be defined over individual traces, such as service level agreements and information-flow properties [15]. In this work, we extend the notion of hyperproperties to cover a broad range of requirements for cyber-physical systems (CPSs), and we present a taxonomy of hyperproperties used to address security and control design concerns for CPSs. Also, we provide practical techniques for automating the process of testing hyperproperties for CPSs.

In contrast to trace properties expressed over individual execution traces, hyperproperties are defined over multiple execution traces. For example, one execution of a system cannot be checked against a service level agreement property such as *"the average time elapsed between a user's request and response over all executions should be less than 1 second"*; the property can only be evaluated over all system execution traces. Moreover, we can consider an information-flow policy of *noninterference* specified as *"for all pairs of traces of a system that have the same low-level security inputs, they will also have the same low-level security output"* [22, 41]. This noninterference property is a hyperproperty as it is expressed over all pairs of traces of a system.

Hyperproperties generalize more traditional formal properties by specifying relationships between disparate execution traces, instead of behaviors of individual execution traces. Traditional logics that consider traces individually, such as LTL, cannot be used to specify hyperproperties, and thus, hyperproperties are more expressive. Logics such as CTL and CTL* allow properties over multiple paths of a computation tree, but they do not permit comparisons between the paths themselves. Instead, to express and efficiently check hyperproperties, Clarkson et al., introduced notions of HyperLTL and HyperCTL* [14]. Both logics directly extend LTL and allow us to reason about more than one execution trace at a time. The main difference between HyperLTL and HyperCTL* is that the former requires trace quantifiers appearing at the beginning of a formula, but the latter allows us to specify them within a formula.

Although hyperproperties are well studied in the context of security policies for software systems, hyperproperties have not been explored for CPSs. For a CPS that includes stochastic factors such as noise, environment disturbance, or transducer inaccuracies, it is realistic for design engineers to expect that the system has some acceptable performance in a probabilistic sense rather than requiring an absolute performance limit be met for all individual behaviors. Acceptable performances defined over the averages of settling time, overshoot, undershoot, or error bounds cannot be specified and checked using individual execution traces; they must be quantified over all execution traces.

Recently, security-aware function modeling of CPSs has emerged as an important research topic in computer science and system verification. A CPS, which is an integration between cyber and physical subcomponents, can be vulnerable to both cyber-based and physical-based attacks [5, 19, 39, 48]. For instance, consider a

MEMOCODE '17, Vienna, Austria
© 2017 ACM. 978-1-4503-5093-8/17/09…$15.00
DOI: 10.1145/3127041.3127058

modern automobile, which is a complex CPS composed of many computer units such as an Engine Control Unit (ECU), the Transmission Control Module (TCM), and an Electronic Brake Control Module (EBCM), all interacting with the physical world via sensors and actuators. Cyber-based attackers can gain access to the communication channels of a modern automobile through wireless or in-vehicle networks. As a result, attackers can infiltrate an ECU or EBCM to stall the engine or disable the brake system [30, 45]. An alternative method of attack involves gaining physical access to the system, for example by manipulating the signals processed by the sensors (known as sensor *spoofing*), to compromise secure information or to alter system behaviors [5, 46]. Instances of active physical-based attacks include vehicle braking system attacks, where faulty data is injected into the wheel speed sensor of a vehicle to disrupt the braking function [48], and insulin delivery device attacks, where glucose level sensor data is corrupted to compromise the function of the insulin delivery service [31]. A passive physical-based attack, also called a side-channel attack, is based on physically observing the system behavior and using leaked information to gain insights into the system implementation [26, 28, 42]. Some well-known side channel attacks are power analysis attacks [27], timing attacks [29], electromagnetic attacks [43] and differential fault analysis attacks [10].

Designing a safety-critical CPS that is entirely secure from both cyber-based and physical-based attacks is challenging or impossible. A reasonable approach is to iteratively improve a CPS control design using a *falsification* technique. Falsification is an automated best-effort approach to identify system behaviors that violate a given formal specification [40]. The design approach would be to first formally specify safety properties of a CPS that protect the system against possible cyber-based and physical-based attacks using formalisms such as temporal logic and to then iteratively improve the design using falsification, which would automatically identify vulnerabilities in the design. Despite the attractiveness of falsification techniques, attacks for CPSs often need to be defined over multiple execution traces of the system, which is something that cannot be expressed or falsified using existing temporal logics such as LTL, MTL, and STL. Thus we propose an extension to these logics that would be compatible with the appropriate specifications. In this work, we present a study of hyperproperties including stability, security and safety, as applied to CPSs. We introduce several instances of hyperproperties capturing relationships (e.g input-output relationships) between multiple traces of a CPS. We extend the syntax and semantics of STL [17] to specify hyperproperties over dense-time real-valued signals, which results in a new logic called HyperSTL. Basically, we add quantifiers at the beginning of an STL formula to express relationships between multiple traces. We also introduce a testing algorithm based on a fragment of HyperSTL and apply it to find falsifying traces for hyperproperties of industrial Simulink models. Moreover, we provide a discussion on the feasibility of falsifying or verifying various classes of hyperproperties for CPSs.

Related work. The study of hyperproperties for CPSs evaluated in this paper was inspired by the previous work of Clarkson and Schneider, who introduced hyperproperties to express security policies such as secure information flows and service level agreements [15]. In [13], Bryans et. al. presented a general formalization of opacity policies that prevent observers from deducing the truth value of a predicate; those opacity policies require behaviors to be specified over multiple paths of a system. In earlier work [37], McLean showed that some “possibilistic” security properties like restrictiveness [35], noninterference[22] and nondeducibility [49] are closure properties that cannot be expressed by individual execution traces. In [37], those properties are specified with respect to different sets of trace contractors called selective interleaving functions.

Following the introduction of hyperproperties [15], Clarkson et al. introduced HyperLTL and HyperCTL*, which are extensions to existing temporal logics, to express and check classes of information-flow hyperproperties [14]. These logics extended LTL and CTL* by adding the path quantifiers that permit specifications involving multiple paths in the system. Model checking algorithms and complexity of fragments of HyperLTL and HyperCTL* were also given in [14], which were then further exploited and applied to check some classes of information-flow hyperproperties in [41].

Prototype implementations of model checkers for HyperLTL and HyperCTL*, which assume the system is modeled as a Kripke structure, can verify some information-flow hyperproperties of a discrete-time system, but extending that work to check hyperproperties defined over continuous traces is a challenging endeavor. For complex CPS models or for models built in frameworks with proprietary or otherwise obfuscated semantics, such as Simulink®, formal verification of hyperproperties is effectively impossible, as no corresponding Kripke structure may be obtained from those models[1]. Alternatively, an easier but still difficult task is to develop an efficient testing framework, which could be used to check hyperproperties for finite collections of traces or could be used to falsify hyperproperties of a CPS model; this is the contribution of the work presented herein.

In [50], Xu et al. introduced a notion of CensusSTL that utilizes STL by adding an *outer* logic to quantify the number of individual agents of a multiagent system whose behaviors satisfy an *inner* STL formula. CensusSTL is similar to the HyperSTL proposed in this paper; however, the former is only able to specify group behaviors from different components of an individual trace while the latter allows us to express relationships between multiple traces.

The remainder of the paper is organized as follows. Section 2 reviews relevant background. Section 3 introduces several examples of hyperproperties of CPSs including stability, security and safety. Section 4 presents the syntax and semantics of HyperSTL. Section 5 and Section 6 describe the testing algorithm for two fragments of HyperSTL. Section 7 applies the proposed approach to find falsifying traces for some hyperproperties of industrial Simulink models, and Section 8 concludes the paper.

[1] Some have created their own translation of Simulink models to modeling languages with well-defined formal semantics (for example, see [3, 52]), but these translations necessarily only handle a subset of the Simulink/Stateflow modeling language. This is due to the fact that some Simulink constructs correspond to behaviors that cannot be modeled using standard frameworks for hybrid systems. One such construct is the Variable Transport Delay block, which, roughly speaking, corresponds to a delay differential equation, a construct that is not handled by standard modeling frameworks for hybrid systems.

2 PRELIMINARIES

In this section, we review the concepts of signal, system, trace property, falsification, and verification.

Signal. We define a signal w as a function $w : \mathbb{T} \to \mathbb{D}$, where $\mathbb{T} \subseteq \mathbb{R}_{\geq 0}$ is the time domain. If $\mathbb{D} = \mathbb{B}$, w is a Boolean signal whose value is either true or false, and if $\mathbb{D} = \mathbb{R}$, then we say that the signal is real-valued. A *trace*, $\mathbf{w} : \mathbb{T} \to \mathbb{D}_1 \times \ldots \times \mathbb{D}_n$, is a collection of n signals, where $\forall t \in \mathbb{T}, \mathbf{w}(t) \overset{\Delta}{=} (w_1(t), w_2(t), ..., w_n(t))$. Intuitively, we can consider $\mathbf{w}$ as one execution trace of a continuous-time system with n variables that describes an evolution of the system. In what follows, we reserve the use of bold letters like $\mathbf{w}$, $\mathbf{w}'$ for traces (i.e., tuples of signals), while we use lowercase italicized letters such as w_i to represent signals.

System. We define a deterministic or nonstochastic[2] cyber-physical system Σ as a function mapping a given input trace in $(\mathbb{T} \to \mathbb{D}^m)$ to an output trace in $(\mathbb{T} \to \mathbb{D}^n)$. We denote by $[\![\Sigma]\!]$ the set of traces $\mathbf{w}$ such that the first m components of $\mathbf{w}$ correspond to the m input signals for $[\![\Sigma]\!]$, and the next n components correspond to the n output signals.

Trace properties. A trace property φ is a finite or infinite set of individual traces. A trace property is either satisfied or violated by any given set of traces [6, 41]. A set of traces W satisfies the trace property φ if $W \subseteq \varphi$. As noted above, an individual trace can have several components, for example, a trace could contain m input signals and n output signals of a given system Σ. We say that the trace property φ holds for a system Σ (denoted as $\Sigma \models \varphi$) if the set of input-output traces compatible with the system description is contained in the trace property, i.e., $[\![\Sigma]\!] \subseteq \varphi$.

Falsification. Given a trace property φ and a CPS Σ, the falsification problem is to find a non-empty set $W \subseteq [\![\Sigma]\!]$ such that $W \nsubseteq \varphi$.

Verification. Given a trace property ϕ, the verification problem of a CPS Σ with respect to ϕ is to show that $[\![\Sigma]\!] \subseteq \phi$.

3 HYPERPROPERTIES OF REAL-VALUED SIGNALS

Hyperproperties generalize formal properties of a system by considering sets of sets of execution traces, instead of only sets of execution traces.

Definition 3.1 (Hyperproperty). Let S denote the set of all traces. Let the power set of S be written as $P \overset{\Delta}{=} \mathcal{P}(S)$. A *hyperproperty* is any subset of $\mathcal{P}(S)$.

We say a set of traces W satisfies a hyperproperty $\phi \subseteq P$ if $W \in \phi$. Given a hyperproperty ϕ and a system Σ, the falsification task is to find a non-empty set $W \subseteq [\![\Sigma]\!]$ such that $W \notin \phi$. Similarly, given a hyperproperty ϕ and a system Σ, the verification task is to show that $[\![\Sigma]\!] \in \phi$.

In this section, we introduce hyperproperties for deterministic systems to characterize properties such as security, safety, and stability. We focus on a class of hyperproperties capturing relationships (*e.g.*, the input-output relationship) between multiple traces of a system, and we show several examples of hyperproperties related to stability and security for CPSs. In rest of this section, we use $d_{sup}(\mathbf{w}, \mathbf{w}')$ to denote the sup-norm distance between traces $\mathbf{w}$ and $\mathbf{w}'$, where $d_{sup}(\mathbf{w}, \mathbf{w}') = \sup_{t \in \mathbb{R}_{\geq 0}} ||\mathbf{w}(t) - \mathbf{w}'(t)||$.

- *Robust behavior* is a requirement that guarantees that small differences in system inputs result in small differences in system outputs. Consider the following property: "*For all pairs of traces of a system with an input difference less than ϵ_1, the output difference should be bounded by ϵ_2*". Such a property is a hyperproperty as it requires at least two execution traces to check. This hyperproperty can be formally written as:

$$\phi_1 \overset{\Delta}{=} \{W \in P \mid \forall \mathbf{w}, \mathbf{w}' \in W : d_{sup}(\mathbf{w}_{in}, \mathbf{w}'_{in}) \leq \epsilon_1 \implies d_{sup}(\mathbf{w}_{out}, \mathbf{w}'_{out}) \leq \epsilon_2\}. \tag{1}$$

  This type of property is related to certain stability notions, such as bounded input, bounded output (BIBO) stability and the $\mathcal{L}_2$ gain, as these notions also bound the variation in the output, based on bounded variation in the input. We note, however, that the robust behavior hyperproperty differs from BIBO stability and the $\mathcal{L}_2$ gain, as the robust behavior hyperproperty is specified over all pairs of execution traces while the BIBO and $\mathcal{L}_2$ properties are defined based on individual traces. The robust behavior hyperproperty is also related to *bisimulation relations* [18] and *conformance-closeness* [2] for a dynamical system, as all three of these properties are based on some constraints on the distances between multiple traces. In fact, we may specify bisimulation or conformance-closeness functions in terms of hyperproperties. Lastly, we note that the robust behavior hyperproperty is perhaps most closely related to Lipschitz Robustness of systems [23], which bounds differences in output behaviors based on bounded differences in input behaviors, though Lipschitz Robustness was originally developed for timed input/output systems as opposed to general CPS models.

- *Side-channel attacks* are attacks against cryptographic devices based on studying leaking information about the operations they process, such as power consumption, heat generation, and execution time. The side channel attack is an instance of an inactive physical-based attack that can be used against a CPS in which some physical behaviors are observable. Attackers can deduce the working principle of a system without either access to the system itself or an understanding of the internal operation of the system. For example, attackers can analyze an abnormal change in the power consumption of an integrated circuit while an encryption process is being executed and then reconstruct the encryption key to access secret data [27, 28]. The following property permits side-channel attacks:

$$\phi_2 \overset{\Delta}{=} \{W \in P \mid \exists \mathbf{w} \in W : \forall \mathbf{w}' \in W : (d_{sup}(\mathbf{w}, \mathbf{w}') > 0 \wedge Power(\mathbf{w}(t)) > c_1) \implies Power(\mathbf{w}'(t)) < c_2\}, \tag{2}$$

  where $Power(\mathbf{w}(t))$ represents the power consumption corresponding to $\mathbf{w}$ over time, and c_1, c_2 are arbitrary constants such that $c_1 > c_2$. A system that satisfies this property allows an

[2] Note the contrast with *stochastic* systems. In stochastic systems, one or more parts of the system have randomness associated with them; for instance, the value of a particular system parameter may be drawn from a probability distribution. The key difference is that the stochastic system may not produce the same output for a given input. Unless otherwise specified, all the systems that we consider in this paper are deterministic.

attacker to detect that a particular behavior has occurred ($\mathbf{w}$ in Formula 2) by monitoring the power associated with the behavior. The property is a hyperproperty as it is expressed in terms of multiple traces. To ensure the safety of a system from the power-monitoring attack, the system should satisfy $\neg\phi_2$. We note that other classes of side-channel attacks such as timing attacks, electromagnetic attacks, and differential fault analysis attacks can be specified using properties similar to Formula 2.

- *Robust control invariance* is a property that can be used to synthesize safe controllers, or more to the point, can be utilized to determine whether a safe controller exists for systems with disturbances [11]. Informally, the property states that, for a given set of behaviors that is deemed safe, a control action exists, such that the system remains within the safe set for any allowable disturbance input. This can be stated formally as follows:

$$\phi_3 \triangleq \{W \in P \mid \exists \mathbf{w} \in W : \forall \mathbf{w}' \in W : (\mathbf{w}, \mathbf{w}') \models \phi\}, \tag{3}$$

where $(\mathbf{w}, \mathbf{w}') \models \phi$ means that the pair $(\mathbf{w}, \mathbf{w}')$ satisfies some property ϕ. In this formulation, $w_u(t)$ is the component of $\mathbf{w}$ that represents the controller action, $w_d(t)$ is a disturbance input, $w_y(t)$ is a system output, and $(\mathbf{w}, \mathbf{w}') \models \phi$ enforces both that $w_u = w'_u$ and $w'_y(t) \in \Omega$, where Ω is the set of safe behaviors. The robust control invariance property is related to fault data injection (FDI) attacks, which are active physical-based attacks where attackers try to input faulty data into a system to corrupt the behavior of the controller. For example, attackers can spoof the sensors of DC microgrids by injecting false data such as the past outputs of the sensors at previous time instants. This instance of FDI attack is also well known as a *replay* attack [8, 31, 48]. FDI attacks have been studied widely for CPS, and many techniques have been proposed to efficiently detect those attacks in the early stages [8, 32, 34]. However, the optimal solution is to design a system that can defend itself against FDI attacks [38]. To guarantee that a system can defend against a sensor attack, given a specification ϕ, it must be possible to choose a controller that ensures that the output of the system always satisfies ϕ, i.e. ϕ_3 must hold.

3.1 Beyond Hyperproperties?

A hyperproperty is more expressive than a trace property as it is defined over a set of sets of traces and requires multiple traces to check. If a system is modeled as trace sets, one interesting question to ask is whether there are system properties inexpressible as hyperproperties. For security policies, all properties of trace sets can be considered as hyperproperties, so the answer may be *negative* [6, 15]. For CPSs, there may exist some properties that are challenging to classify.

Consider the following property specifying the *Lyapunov stability* of a dynamical control system:

$$\phi_{Ly} \triangleq \{\forall \epsilon \in [0, \infty), \exists \delta \in [0, \epsilon), \forall \mathbf{w} \in W : \\ ||\mathbf{w}(0)|| < \delta \implies (t > 0 \wedge ||\mathbf{w}(t)|| < \epsilon)\}. \tag{4}$$

Intuitively, this property indicates that a system is Lyapunov stable if for any ϵ-*ball* around the origin, there exists a δ-*ball* around the origin ($\delta < \epsilon$) such that if the system starts within the δ-*ball*, then it will never leave the ϵ-*ball* [9]. The illustration of a Lyapunov stable system is shown in Figure 1.

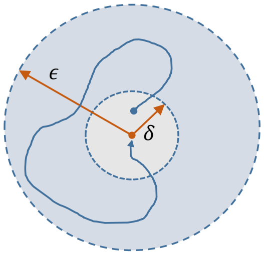


Figure 1: Illustration of a Lyapunov stable system.

Lyapunov stability is specified over the space of parameters and execution traces, and involves two alternations between universal and existential quantifiers. As we cannot check the Lyapunov stability with individual traces, it is not a trace property; so is it a hyperproperty? Consider the parameters δ and ϵ as constant signals, and then rewrite Lyapunov stability as follows:

$$\phi'_{Ly} \triangleq \{W \in P \mid \forall \mathbf{w} \in W \ : \exists \mathbf{w}' \in W \ : \forall \mathbf{w}'' \in W : \\ ||w''_{out}(0)|| < w'_{\delta}(0) \implies (t > 0 \wedge ||w''_{out}(t)|| < w_{\epsilon}(t))\}, \tag{5}$$

where a trace $\mathbf{w}$ is composed of two constant input signals w_δ, w_ϵ and an output signal w_{out}. By mapping parameters into constant signals, we can express interesting properties of the system as hyperproperties. Then Lyapunov stability is a hyperproperty that requires multiple traces to check; and it can be formally specified using the HyperSTL introduced in the next section. As to the original question of whether all system properties of interest can be specified as hyperproperties, we leave this open.

Remark 3.2 Although we focus on describing hyperproperties defined over real-valued signals, we note that there are other hyperproperties that can be specified in the context of CPSs as well. For instance, the *nondeducibility property* is an important information-flow security policy that prevents a low-level observer with sufficient knowledge of a target CPS from deducing high-level (confidential) information. The nondeducibility property is defined such that for each low-level input trace, there are more than one possible high-level input traces that produce the same output. Intuitively, an attacker should not be able to distinguish between permissible high-level behaviors based on low-level behaviors [20, 36]. On the other hand, the *noninterference property* is another important information-flow security policy that requires that high-level security users should not interfere with low-level security users. Intuitively, the outputs observed by the low-level security users remain unchanged despite the actions of the high-level security users [22]. Other variants of the noninterference property such as *noninference* [37], *observational determinism* [51], *declassification* [44], and *quantitative nonterinference* [47] are also hyperproperties that need to be specified over multiple traces. Though the nondeducibility and noninterference properties are relevant for CPS, in many cases their impact on and from real-valued signals is tenuous, and so we do not treat them further herein.

4 HYPERSTL

In this section, we introduce HyperSTL, a temporal logic that can be used to specify a class of hyperproperties of real-valued signals. The syntax and semantics of HyperSTL are naturally extended from those of STL by adding existential and universal trace quantifiers into STL's syntax to relate multiple execution traces [17].

Syntax. Let $\mathbf{v}$ be a *trace variable* from an infinite set of trace variables $\mathcal{V}$. The syntax of HyperSTL is then defined as follows:

$$\begin{aligned} \phi &:= \exists \mathbf{v}.\phi \mid \forall \mathbf{v}.\phi \mid \varphi \\ \varphi &:= true \mid \mu_{\mathbf{v}} \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \mathbf{U}_I \varphi \end{aligned}$$

Here, we add a universal quantifier $\forall$ and an existential quantifier $\exists$ to the syntax to indicate whether we want to specify that a formula holds over all traces or over at least one trace, respectively. For instance, $\forall \mathbf{v}.\exists \mathbf{v}'.\phi$ means that for any trace $\mathbf{w}$ assigned to trace variable $\mathbf{v}$, there exists a trace $\mathbf{w}'$ that can be assigned to trace variable $\mathbf{v}'$ such that ϕ holds on these two traces. We define $\Pi : \mathcal{V} \rightarrow S$ as a *trace assignment* (i.e., a valuation), which is a partial function mapping trace variables to traces, and S is a set of all infinite traces. Let v_i be the projection of a trace variable $\mathbf{v}$ along its i^{th} component, the projection of a trace assignment $\Pi(v_i)$ maps v_i to the i^{th} component of a trace $\mathbf{w}$ (i.e., w_i). Also, we abuse the subscript notation of a trace's component to write its corresponding trace variable's component in a HyperSTL formula, e.g., w_{out} is represented by v_{out}. A trace $\mathbf{w}$ can be Booleanized through atomic predicates of the form $\mu_{\mathbf{w}} \triangleq f(w_1(t), w_2(t), ..., w_n(t)) > 0$, where f is a real-valued function. Then, $\mu_{\mathbf{v}} = f(\Pi(\mathbf{v})(t)) > 0$ represents a Booleanized atomic predicate $\mu_{\mathbf{w}}$ if $\mathbf{v}$ is instanced by $\mathbf{w}$. Also, I is an interval over $\mathbb{R}_{\geq 0}$ such as $[a, b)$, (a, b), $(a, b]$, $[a, b]$, $(a, +\infty)$, or $[a, +\infty)$, where a, b are real numbers and $0 \leq a < b$. If I is not specified, we assume that $I = [0, \infty)$. We also allow Boolean operators $\vee$ and $\implies$ with their standard meaning. Temporal operators used in HyperSTL formulas include *always* ($\Box$), *eventually* ($\Diamond$), and *until* ($\mathbf{U}$), respectively, where $\Diamond_I \varphi = true\mathbf{U}_I \varphi$, and $\Box_I \varphi = \neg\Diamond_I \neg\varphi$. Note that we use trace variables such as $\mathbf{v}$, $\mathbf{v}'$ to express HyperSTL formula and the corresponding traces represented by these trace variables like $\mathbf{w}$ $\mathbf{w}'$ to interpret the formula. Consider the HyperSTL formula $\phi := \exists \mathbf{v}.\forall \mathbf{v}'.\Box_{[0,1]}(||\mathbf{v} - \mathbf{v}'|| < 1)$. This property says that there is always a trace $\mathbf{w}$, such that for all times in the interval $[0, 1]$, every other trace $\mathbf{w}'$ is at a bounded distance of 1 from $\mathbf{w}$.

Boolean Semantics. A HyperSTL formula satisfied by a set of traces W at a time t is written as $\Pi, t \models_W \phi$, The validity judgment of a HyperSTL formula at a given time t is specified according to the following recursive semantics:

$$\begin{array}{rcl} \Pi, t \models_W \exists \mathbf{v}.\phi & \text{iff} & \text{exists } \mathbf{w} \in W \;:\; \mathbf{w} \models \phi \text{ and } \Pi(\mathbf{v}) = \mathbf{w} \\ \Pi, t \models_W \forall \mathbf{v}.\phi & \text{iff} & \text{forall } \mathbf{w} \in W \;:\; \mathbf{w} \models \phi \text{ and } \Pi(\mathbf{v}) = \mathbf{w} \\ \Pi, t \models_W \mu_{\mathbf{v}} & \text{iff} & f(\Pi(\mathbf{v})(t)) > 0 \\ \Pi, t \models_W \neg\varphi & \text{iff} & \Pi, t \not\models_W \varphi \\ \Pi, t \models_W \varphi_1 \wedge \varphi_2 & \text{iff} & \Pi, t \models_W \varphi_1 \text{ and } \Pi, t \models_W \varphi_2 \\ \Pi, t \models_W \varphi_1 \mathbf{U}_I \varphi_2 & \text{iff} & \exists t_1 \in t + I \text{ s.t } \Pi, t_1 \models_W \varphi_2 \\ & & \text{and } \forall t_2 \in [t, t_1] \text{ s.t } \Pi, t_2 \models_W \varphi_1 \end{array}$$

Using HyperSTL, we can express the hyperproperties described in Section 3 over some time interval $[t_1, t_2]$ as follows[3].

- The robust behavior in Formula 1 can be specified as:

$$\begin{aligned} \phi'_1 \triangleq \forall \mathbf{v}.\forall \mathbf{v}'.\, \Box_{[t_1,t_2]}(d_{sup}(\mathbf{v}_{in}, \mathbf{v}'_{in}) \leq \epsilon_1 \\ \implies d_{sup}(\mathbf{v}_{out}, \mathbf{v}'_{out}) \leq \epsilon_2). \end{aligned} \tag{6}$$

- The power-monitoring attack in Formula 2 can be written as:

$$\begin{aligned} \phi'_2 \triangleq \exists \mathbf{v}.\forall \mathbf{v}'.\, \Box_{[t_1,t_2]}((d_{sup}(\mathbf{v}, \mathbf{v}') > 0 \\ \wedge\, Power(\mathbf{v}) > c_1) \implies Power(\mathbf{v}') < c_2). \end{aligned} \tag{7}$$

Furthermore, we can rewrite the Lyapunov stability specified in Formula 5 as the following HyperSTL formula

$$\phi''_{Ly} \triangleq \forall \mathbf{v}.\exists \mathbf{v}'.\forall \mathbf{v}''.\, (v''_{out} < v'_{\delta} \implies \Box_{(0,\infty)} v''_{out} < v_{\epsilon}). \tag{8}$$

According to the possible alternation of quantifiers in a HyperSTL's syntax, we classify the above HyperSTL formulae into two fragments:

(a) *alternation-free* HyperSTL formulae including one type of quantifier, and
(b) *k-alternation* HyperSTL formulae that have k number of alternations between existential and universal quantifiers.

Thus, the robust behavior property can be expressed using alternation-free HyperSTL while the power-monitoring attack property can be specified using 1-alternation HyperSTL. The Lyapunov stability property is more complex as it must be expressed using 2-alternation HyperSTL.

Falsification or Verification of Hyperproperties? We have introduced several classes of hyperproperties for CPSs and a temporal logic approach to express them. Next, we investigate whether we can falsify or verify those hyperproperties using existing methods. Hyperproperties are more complex and expressive than traditional properties, and performing falsification and verification for hyperproperties is harder, in many cases. Despite this, we observe that certain classes of hyperproperties can be falsified or verified. For instance, we can falsify an alternation-free HyperSTL formula that contains a universal quantifier (e.g., the robust behavior hyperproperty), and we can verify an alternation-free HyperSTL formula that contains an existential quantifier. For the class of hyperproperties that includes alternating quantifiers, falsification or verification are often undecidable unless we impose some assumption about the sets of execution traces (e.g., quantified over some finite set of traces with bounded time).

4.1 t-HyperSTL

We introduce t-HyperSTL as a fragment of HyperSTL in which a nesting structure of temporal logic formulas involving different traces is not allowed. For example, a formula $\forall \mathbf{v}.\exists \mathbf{v}'.\Box_{[0,2]}\mathbf{v} > 1 \implies \Diamond_{[1,2]}\mathbf{v}' > 2$ is allowed but a formula $\forall \mathbf{v}.\exists \mathbf{v}'.\Box_{[0,2]}(\mathbf{v} > 1 \implies \Diamond_{[1,2]}\mathbf{v}' > 2)$ is not allowed. Also, t-HyperSTL restricts the *until* operator to be specified over an individual trace, e.g., t-HyperSTL does not allow the formula $\forall \mathbf{v}.\exists \mathbf{v}'.(\mathbf{v} > 1)\mathbf{U}_{[0,1]}(\mathbf{v}' > 2)$.

Inherited from the syntax of HyperSTL, t-HyperSTL formulae are also classified into alternation-free and k-alternation types.

[3] For a robust control invariance hyperproperty, an instance of the corresponding HyperSTL formula will be shown in Section 7.2.

t-HyperSTL suffices to express the class of hyperproperties formulated in Section 3, and its corresponding semantics, which is more restrictive than that of HyperSTL, allow us to perform falsification for these hyperproperties.

Quantitative Semantics. The quantitative semantics of t-HyperSTL reflects the *robustness satisfaction* of a t-HyperSTL formula. It is a natural extension of those for STL [17, 33]. Given χ is a real-valued function of a formula φ, a trace assignment Π, a trace variable $\mathbf{v}$, and a time t, the quantitative semantics of t-HyperSTL is defined inductively as follows:

$$\begin{aligned}
\chi(\varphi, \Pi, \exists \mathbf{v}, t) &= \max_{\mathbf{w} \in W} \chi(\varphi, \Pi(\mathbf{v}) = \mathbf{w}, t) \\
\chi(\varphi, \Pi, \forall \mathbf{v}, t) &= \min_{\mathbf{w} \in W} \chi(\varphi, \Pi(\mathbf{v}) = \mathbf{w}, t) \\
\chi(\mu_{\mathbf{v}} > 0, \Pi, \mathbf{v}, t) &= \mu_{\mathbf{v}} \\
\chi(\neg\varphi, \Pi, \mathbf{v}, t) &= -\chi(\varphi, \Pi, \mathbf{v}, t) \\
\chi(\varphi_1 \wedge \varphi_2, \Pi, \mathbf{v}, t) &= \min(\chi(\varphi_1, \Pi, \mathbf{v}, t), \chi(\varphi_2, \Pi, \mathbf{v}, t)) \\
\chi(\varphi_1 \mathrm{U}_I \varphi_2, \Pi, \mathbf{v}, t) &= \sup_{t_1 \in t+I} \min(\chi(\varphi_2, \Pi, \mathbf{v}, t_1), \\
&\qquad \inf_{t_2 \in [t, t_1]} \chi(\varphi_1, \Pi, \mathbf{v}, t_2))
\end{aligned}$$

5 FALSIFYING ALTERNATION-FREE T-HYPERSTL

We first consider the falsification of alternation-free t-HyperSTL formulae. This fragment of HyperSTL is expressive enough to capture a broad range of hyperproperties specifying input-output relationships over all pairs of execution traces. We use a translation scheme called self-composition [7], which allows us to falsify an alternation-free t-HyperSTL formula that includes only universal quantifiers using a robust testing method for a normal STL formula. Then, given an alternation-free t-HyperSTL that includes universal quantifiers, we attempt to find a set of falsifying traces for CPSs corresponding to this formula.

Falsification algorithm. The procedure that addresses the falsification problem of a system Σ with respect to a given hyperproperty φ_h over a time duration T is shown in Algorithm 1, and further interpreted as follows.

- We first transform the alternation-free t-HyperSTL formula φ_h into the equivalent STL formula φ_{STL}.
- We then call a function NewSystemGen to generate a new model that contains copies of the original system. The number of copies is equal to the number of quantifiers of the formula φ_h.
- Then, we apply existing falsification mechanisms for an STL formula such as Breach[4] [16] to compute the minimum robustness value χ_{min} of the system Σ' according to φ_{STL}. Breach allows us to parametrically generate different input signals over a parameter space. For example, parameters can represent control points, and an input signal can be created using interpolation between these points. If χ_{min} is negative we return the optimal set of parameters $\Theta_f \in \Theta$ that produces a falsifying behavior.

Algorithm 1 Falsification of alternation-free t-HyperSTL

```
Require: a system $\Sigma$, a parameter space $\Theta$,
a t-HyperSTL formula $\varphi_h$, a time duration $T$,
a maximum number of simulations N
begin
  $\varphi_{STL}$ ← HyperSTL2STL($\varphi_h$) // transform specification
  $\Sigma'$ ← NewSystemGen($\Sigma$, $\varphi_h$) // transform model
  $\chi_{min}, \Theta_f$ ← FalsifySTL($\Sigma'$, $\varphi_{STL}$, $\Theta$, $T$, $N$)
  if $\chi_{min} < 0$ then
    return $\Theta_f$
  end
end
```

We note that, unlike formal verification, performing falsification cannot ensure a system is always safe; even if falsification fails to identify a falsifying behavior, a counter-example may still exist.

Example 5.1. Consider a mechanical mass-spring damper system whose dynamics are defined by the second-order ordinary differential equation:

$$\ddot{x}(t) + 2\dot{x}(t) + 5x(t) = 3F(t), \tag{9}$$

where x is the vertical position of the mass, and F is the random external force. The robust behavior hyperproperty of the system is specified as follows: for all pairs of traces of the system with the external force difference less than ϵ_1 , the output difference should be bounded by ϵ_2; here $\epsilon_1 = 0.2$ and $\epsilon_2 = 0.3$. We apply the Algorithm 1 to falsify the robust behavior hyperproperty for the system with a duration $T = 10$ seconds. Formula 6 can be reduced to the normal STL formula as follows:

$$\phi_M \overset{\Delta}{=} \square_{[0,10]}(\rho_{in} \leq \epsilon_1 \implies \rho_{out} \leq \epsilon_2), \tag{10}$$

where a trace $\mathbf{p} \overset{\Delta}{=} (\rho_{in}, \rho_{out})$ of the system Σ' captures the input-output difference between two traces $\mathbf{w}, \mathbf{w}'$ of the original system Σ', e.g., $\rho_{in}(t) = ||w_{in}(t) - w'_{in}(t)||$. Here, the system Σ' contains two copies of the mechanical mass-spring damper system Σ. The falsification result shown in Figure 2 illustrates the inductive checking procedure for the satisfaction of Formula 10 using Breach, where $alw_{[0,10]}$ is equivalent to $\square_{[0,10]}$, and the left y-axis denotes robustness degree. Here, we observe that the violation of the robust behavior hyperproperty of the mechanical mass-spring damper system occurs during the overshoot period of the outputs of the system.

Remark 5.2 There is a duality between addressing the falsification problem of an alternation-free t-HyperSTL that only contains universal quantifiers and solving the verification problem of an alternation-free t-HyperSTL that only contains existential quantifiers. Given an alternation-free t-HyperSTL such as $\exists\mathbf{v}.\exists\mathbf{v}'.\phi_e$, our purpose is to extensively simulate a system and find a single pair of execution traces of the system that satisfies ϕ_e. Here, we do not attempt to falsify the system, but verify the system. Thus, this process is dual to finding the falsifying traces of the system corresponding to the formula $\forall\mathbf{v}.\forall\mathbf{v}'.\neg\phi_e$.
Also, we note that we can leverage Algorithm 1 such that it includes a parameter synthesis approach to mine hyperproperties for CPSs, as in [24, 25]. For instance, we could use a requirement mining

[4] Breach [16] is a tool that applies a best-effort approach to automatically check whether a system satisfies a given STL formula.

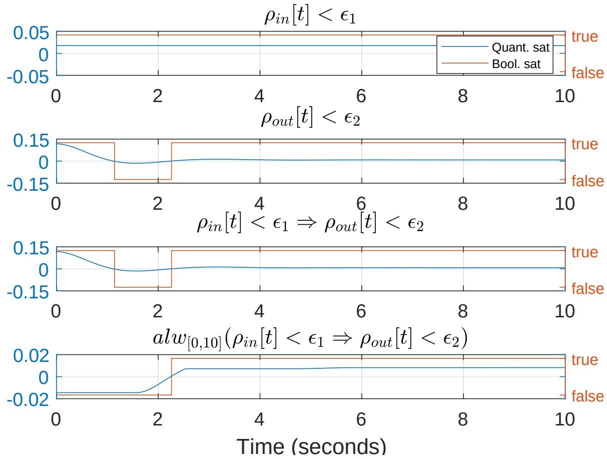


Figure 2: Falsification result of the mass-spring damper system. The counterexample pair of traces found by Breach for the robust behavior hyperproperty.

approach to automatically infer appropriate values for the ϵ_1 and ϵ_2 variables in Formula 10.

6 FALSIFYING K-ALTERNATION T-HYPERSTL

Falsifying k-alternation t-HyperSTL formulas is a challenging task, as it requires us to examine all execution traces of a system. Consider a 1-alternation t-HyperSTL formula such as $\exists \mathbf{v}.\forall \mathbf{v}'.\phi$; falsifying a system for this property is as hard as verifying the system, since we need to show that for all traces $\mathbf{w} \in S$, there exists a trace $\mathbf{w}'$ that the formula ϕ is violated, where S is an infinite set of traces. It is even more difficult to perform falsification for CPSs whose dynamics evolve continuously over time. Furthermore, if a hyperproperty contains more than one alternation of quantifiers (e.g. the Lyapunov stability property), the falsifying algorithm may suffer an exponential growth in complexity. Despite this, if we assume a CPS can be modeled by a finite set of traces, we can develop a falsifying algorithm for the system that can prove or disprove ϕ.

In general, there may not exist a unique answer to the question of whether we can verify or falsify a system with respect to the formula $\exists \mathbf{v}.\forall \mathbf{v}'.\phi$ using finite simulations. We can consider several possible answers for that question as follows.

- **Case 1**: if both $\mathbf{w}, \mathbf{w}'$ belong to some infinite set of traces, then we can neither verify nor falsify ϕ.
- **Case 2**: if $\mathbf{w}$ belongs to an infinite set of traces and $\mathbf{w}'$ belongs to a finite set of traces, then we cannot falsify but we can verify ϕ.
- **Case 3**: if $\mathbf{w}$ belongs to a finite set of traces and $\mathbf{w}'$ belongs to an infinite set of traces, then we cannot verify but we can falsify ϕ.
- **Case 4**: If both $\mathbf{w}$ and $\mathbf{w}'$ belong to a finite set of n traces, we are able to verify the system with n simulations as well as falsify the system with $\frac{n(n-1)}{2}$ simulations.

We note that in all of the cases that we are able to falsify the system corresponding to the formula $\exists \mathbf{v}.\forall \mathbf{v}'.\phi$ with finite simulations, we can apply Algorithm 1 to transform the falsification problem to another equivalent problem that uses a traditional STL specification.

Table 1: Feasibility of solving the falsification and verification problems for properties and hyperproperties expressed using STL and k-alternation t-HyperSTL under two assumptions: A1) using finite simulation and A2) applying a verification oracle that can do reachability analysis with respect to the last quantifier.

Type	A1: Finite Simulation		A2 : Verification Oracle
	Falsification	Verification	on the Last Quantifier
∀	Yes	No	-
∃	No	Yes	-
∀∃	No	No	∀
∃∀	No	No	∃
∀∃∀	No	No	∀∃
∃∀∃	No	No	∃∀

The falsification procedure is similar to solving the falsification problem of alternation-free t-HyperSTL.

For the case that both execution traces of a system, $\mathbf{w}$ and $\mathbf{w}'$, belong to some infinite sets, and if we have a verification oracle to address the last quantifier (e.g., by conservatively estimating the set of possible system behaviors, under certain conditions), we can either falsify or verify the system. Given a set of initial states, a verification oracle can be a method that mathematically overapproximates the reachable set of the system or a simulation-based technique [1, 21] that may verify the system with finite simulations.

Alternatively, for a hyperproperty that requires two or more alternations of quantifiers to express, even if we have a verification oracle corresponding to the last quantifier, we can neither falsify nor verify a system. Using a verification oracle, the feasibility of addressing the falsification and verification problems associated with a k-alternation t-HyperSTL formula is equivalent to that of a $(k-1)$-alternation t-HyperSTL formula; this is shown in Table 1. We emphasize that any hyperproperties for general CPSs that are as complex as, or more complicated than Lyapunov stability, are not verifiable or falsifiable without reasonable restrictions on sets of execution traces.

7 CASE STUDY

In this section, we introduce two proof-of-concept case studies in the domain of automotive control systems: a) an industrial-scale Simulink model of a closed-loop airpath control (APC) system and b) a Simulink model of a fault-tolerant fuel (FTF) control system. We will demonstrate how to apply the testing framework of HyperSTL built on top of Breach to falsify the robust behavior hyperproperty of the APC system, and the robust control invariance hyperproperty of the FTF system under FDI attacks.

7.1 Airpath Control Model

We use a prototype APC system to evaluate the capability of our proposed method on an industrial control system. The APC is a key subsystem for a hydrogen Fuel-Cell (FC) vehicle powertrain. The

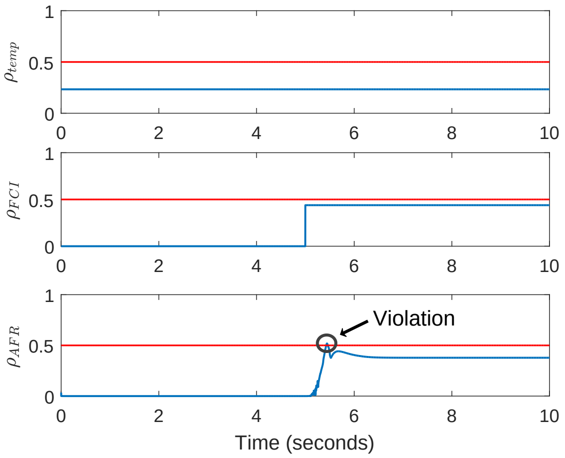


Figure 3: Falsification result of the APC system. The counterexample pair of traces found by Breach for the robust behavior hyperpropperty.

purpose of the APC is to regulate the air flow rate into the FC stack using multiple actuators. The FC stack generates electrical power for the vehicle using a mixture of air and hydrogen. The FC stack only operates under restricted conditions, such as temperature, pressure and moisture level within the stack. An excess of moisture in the stack will impede the performance while moisture deficiency could permanently damage the FC stack. Thus, to achieve high performance while still operating the system in a safe regime, the controller is required to accurately regulate the air flow rate.

The closed-loop Simulink model of the APC system is complex; it contains more than 7,000 Simulink blocks such as integrators, saturations, S-Function blocks, lookup tables, and data store memory blocks. The model has two input signals including i) the ambient temperature and ii) the fuel cell current request (FCI). Details of the system, such as units and expected signal ranges, are suppressed due to proprietary concerns. Intuitively, an FCI value is proportional to the desired torque requested by the driver, which is ultimately based on the accelerator pedal angle. The output of the APC system is an air flow rate (AFR). The purpose of the controller model is to regulate the AFR to some desirable reference value. To ensure the APC system works properly, for some small perturbations of the ambient temperature and FCI values, the differences in AFR values should be bounded within a desirable range. In other words, to avoid unexpected changes in the air flow rate at the inlet of an FC stack, which may cause undesirable behavior, the system should satisfy the robust behavior hyperproperty. The robust behavior hyperproperty of the APC system can be formalized as follows,

$$\begin{aligned}\phi_{APC} \triangleq \{W \in P \mid \forall \mathbf{w}, \mathbf{w}' \in W :\\ (d_{sup}(w_{temp}, w'_{temp}) \leq \epsilon_1 \wedge d_{sup}(w_{FCI}, w'_{FCI}) \leq \epsilon_2)\\ \implies d_{sup}(w_{AFR}, w'_{AFR}) \leq \epsilon_3)\},\end{aligned} \tag{11}$$

which can be translated to the following STL formula using Algorithm 1 to perform the falsification task,

$$\phi'_{APC} \triangleq \square_{[0,T]}((\rho_{temp} \leq \epsilon_1 \wedge \rho_{FCI} \leq \epsilon_2) \implies \rho_{AFR} \leq \epsilon_3), \tag{12}$$

where a trace $\mathbf{w}$ is composed of the temperature and FCI input signals w_{temp} and w_{FCI} respectively, and the AFR output signal w_{AFR}. Here, we create a new model including two copies of the original APC system; and a trace $\mathbf{p} \triangleq (\rho_{temp}, \rho_{FCI}, \rho_{AFR})$ of the new model captures the input-output difference between two traces $\mathbf{w}, \mathbf{w}'$ of the original model, for instance, $\rho_{temp}(t) = ||w_{temp}(t) - w'_{temp}(t)||$.

The result of falsification of the robust behavior hyperpropety of the APC system is shown in Figure 3, where the blue lines present the distance signals ρ_{temp}, ρ_{FCI}, ρ_{AFR} respectively, and the red lines demonstrate their corresponding bounds. Here, the parameter values selected by a design engineer are normalized to 0.5. That is, $\epsilon_1 = 0.5$, $\epsilon_2 = 0.5$, and $\epsilon_3 = 0.5$. The sampling time is 0.001024 seconds and the simulation time T is 10 seconds. For proprietary reasons, we normalize the quantities and suppress the units for the data shown in the figure. The counterexample pairs of traces reported by Breach demonstrate a behavior where the output difference exceeds its allowed bounds when the input differences are still less than their given thresholds, which is a violation of Formula 12. Finding this counter-example is significant, as it can help automotive control engineers to improve the controller design to eliminate such an undesirable behavior of the APC system.

7.2 Fault-tolerant Fuel Model

We consider a fault-tolerant fuel (FTF) model that includes both Simulink blocks and Stateflow charts[5]. The model has two external input signals, engine speed and throttle command, and one output signal, which is the effective air-fuel ratio inside the combustion chamber. The model also contains four sensors measuring throttle angle, engine speed, the amount of residual oxygen in the exhaust gas (EGO), and the manifold absolute pressure (MAP). The controller has three different control strategies: a normal operation mode, which is used when no sensor faults are present, a fault mitigation mode, which is used when one sensor fault has occurred, and a mode that disables fuel control, which is used when two or more sensor faults are detected. We only consider the normal and fault mitigation modes for this example. The goal of the controller is to regulate the air-fuel ratio output, denoted as λ, so that it remains within a desirable range, despite a failure in at most one sensor.

In this case study, we evaluate the ability of the FTF controller to tolerate an engine speed sensor fault. In the original version of the model, a speed sensor fault consists of the speed sensor output being set to 0.0 rad/sec; the controller detects the fault when the sensor reading equals 0.0. In the modified version that we use, we do not fix the controller mode based on the sensor reading, but instead we evaluate the controller performance when either the normal or fault mitigation modes are selected. In the modified version of the model that we use, a speed sensor fault consists of a sensor output producing a fixed but randomly selected value in the sensor range [0, 620] rad/sec. This kind of sensor fault could occur when an attacker uses a sensor spoofing approach to inject incorrect measurements into the sensor readings or when a real fault occurs in the speed sensor. We use the robust control invariance property to specify desired controller performance in the presence of the

[5] We use a modified version of the FTF model available at https://www.mathworks.com/help/simulink/examples/modeling-a-fault-tolerant-fuel-control-system.html

indicated class of sensor faults:

$$\phi_{FTF} \triangleq \exists \mathbf{v}.\forall \mathbf{v}'.\Box_{[\tau,\infty]}(d_{sup}(v_u, v'_u) = 0 \implies 0.8\lambda_{ref} \leq v'_\lambda \leq 1.2\lambda_{ref}), \quad (13)$$

where λ_{ref} is the reference value of the air-fuel ratio λ, and τ is the settling time. Here, a trace variable $\mathbf{v}$ can be mapped to a trace $\mathbf{w}$ composed of the controller input w_u corresponding to a controller mode decision, a disturbance w_d representing the fixed random sensor input injected into the speed sensor, and an output w_λ.In general, we cannot falsify Formula 13 according to the discussion shown in Table 1; however, for systems like the FTF model that have a finite set of control strategies, we can effectively perform falsification by creating a new model that contains copies of the original system, one copy for each control mode (two copies, in this case). The external input (the speed sensor reading) is connected to each of the copies of the model. The specification ϕ_{FTF} is converted to the following equivalent formula in standard STL:

$$\hat{\phi}_{FTF} \triangleq \forall w_d.\Box_{[\tau,\infty]}(0.8\lambda_{ref} \leq w_{\lambda_1} \leq 1.2\lambda_{ref} \vee 0.8\lambda_{ref} \leq w_{\lambda_2} \leq 1.2\lambda_{ref}), \quad (14)$$

where w_{λ_1} and w_{λ_2} are the air-fuel ratios of the first and second copies of the model. We note that Formula 14 is arrived at by applying the quantitative semantics provided in Sec. 4; the disjunction in Formula 14 appears due to the $\exists$ quantifier in Formula 13, which effectively applies a max operator over the two available control modes. The formula $\hat{\phi}_{FTF}$ can be tested using the falsification methods for traditional STL available in Breach.

Figure 4 illustrates the falsification result of the FTF model. The blue lines correspond to a simulation trace representing the falsifying behavior, the green line illustrates an instance of the correct speed, and the red lines represent the error bound of λ, where $\tau = 10$ seconds, $T = 50$ seconds, and $\lambda_{ref} = 14.6$. Based on the results, we can conclude that there exists a trace, which includes outputs w_{λ_1} and w_{λ_2} that both evolve beyond the tolerance bound regardless of whether the controller operates in the normal mode or the fault mitigation mode (i.e., the performance requirement is violated despite which control mode is used). This experiment demonstrates the capability of using a falsification approach to automatically test hyperproperties for CPSs.

8 CONCLUSION AND FUTURE WORK

In this paper, we represented the first study of the hyperproperties of CPSs. We defined a new temporal logic, called HyperSTL, to express several hyperproperties including stability, security, and safety for CPSs. HyperSTL allows us to effectively specify more general requirements of CPS rather than STL as it can express the relationships between multiple execution traces. The testing framework of t-HyperSTL, a fragment of HyperSTL, was also given and applied to falsify the robust behavior hyperproperty of a hydrogen fuel-cell powertrain model, and the robust control invariance hyperproperty of the fuel control model under a fault data injection attack. We also discuss the feasibility of performing the falsification and verification for various classes of hyperproperties for CPSs.

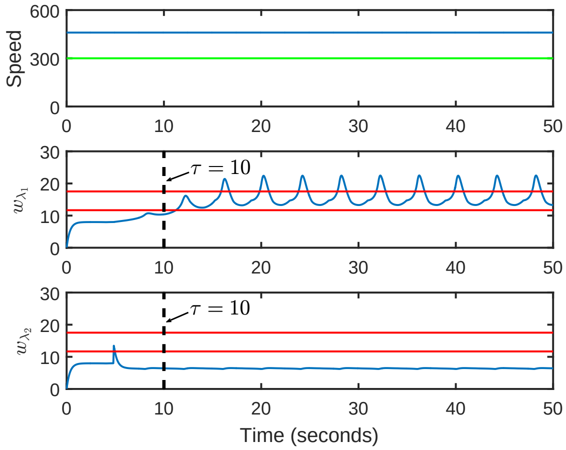


Figure 4: A pair of falsifying traces found by Breach illustrating the FTF model cannot tolerate the fault under a speed sensor fault.

Future Work. We first plan to introduce a library of HyperSTL fomulae that encapsulates different general classes of hyperproperties of CPS including those presented in this paper. Second, the falsification algorithm of HyperSTL proposed in the paper is incomplete as it relies on self-composition (i.e. making copies of a system) and only falsifies a restricted class of hyperproperties. Thus, extending the falsification algorithm to bypass self-composition to falsify more interesting hyperproperties is planned. Also, the monitoring algorithms of HyperLTL recently proposed in [4, 12] could be applied to HyperSTL.

9 ACKNOWLEDGMENTS

The authors would like to acknowledge Borzoo Bonakdarpour and his research group for insightful comments that helped us refine our definition of HyperSTL. We would also like to thank the anonymous reviewers for their feedback. The material presented in this paper is based upon work supported by the National Science Foundation (NSF) under grant numbers CNS 1464311, EPCN 1509804, and SHF 1527398, the Air Force Research Laboratory (AFRL) through contract numbers FA8750-15-1-0105, and FA8650-12-3-7255 via subcontract number WBSC 7255 SOI VU 0001, and the Air Force Office of Scientific Research (AFOSR) under contract numbers FA9550-15-1-0258 and FA9550-16-1-0246. The U.S. government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of AFRL, AFOSR, or NSF.

REFERENCES

[1] Houssam Abbas, Bardh Hoxha, Georgios Fainekos, and Koichi Ueda. 2014. Robustness-guided temporal logic testing and verification for stochastic cyber-physical systems. In *Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2014 IEEE 4th Annual International Conference on*. IEEE, 1–6.

[2] Houssam Abbas, Hans Mittelmann, and Georgios Fainekos. 2014. Formal property verification in a conformance testing framework. In *Formal methods and models for codesign (memocode), 2014 twelfth acm/ieee international conference on*. IEEE, 155–164.

[3] Aditya Agrawal, Gyula Simon, and Gabor Karsai. 2004. Semantic Translation of Simulink/Stateflow Models to Hybrid Automata Using Graph Transformations. *Electronic Notes in Theoretical Computer Science* 109 (2004), 43 – 56.

[4] Shreya Agrawal and Borzoo Bonakdarpour. 2016. Runtime verification of k-safety hyperproperties in HyperLTL. In *Computer Security Foundations Symposium (CSF), 2016 IEEE 29th*. IEEE, 239–252.

[5] Mohammad Al Faruque, Francesco Regazzoni, and Miroslav Pajic. 2015. Design methodologies for securing cyber-physical systems. In *Proceedings of the 10th International Conference on Hardware/Software Codesign and System Synthesis*. IEEE Press, 30–36.

[6] Mack W Alford, Jean-Pierre Ansart, Günter Hommel, Leslie Lamport, Barbara Liskov, Geoff P Mullery, and Fred B Schneider. 1985. *Distributed systems: methods and tools for specification. An advanced course*. Springer-Verlag New York, Inc.

[7] Gilles Barthe, Pedro R D'Argenio, and Tamara Rezk. 2004. Secure information flow by self-composition. In *Computer Security Foundations Workshop, 2004. Proceedings. 17th IEEE*. IEEE, 100–114.

[8] Omar Beg, Taylor Johnson, and Ali Davoudi. 2017. Detection of False-data Injection Attacks in Cyber-Physical DC Microgrids. *IEEE Transactions on Industrial Informatics* (2017).

[9] Nam Parshad Bhatia and Giorgio P Szegö. 2002. *Stability theory of dynamical systems*. Springer Science & Business Media.

[10] Eli Biham and Adi Shamir. 1997. Differential fault analysis of secret key cryptosystems. In *Annual International Cryptology Conference*. Springer, 513–525.

[11] F. Blanchini. 1999. Survey Paper: Set Invariance in Control. *Automatica* 35, 11 (Nov. 1999), 1747–1767.

[12] Noel Brett, Umair Siddique, and Borzoo Bonakdarpour. 2017. Rewriting-Based Runtime Verification for Alternation-Free HyperLTL. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 77–93.

[13] Jeremy W Bryans, Maciej Koutny, Laurent Mazaré, and Peter YA Ryan. 2008. Opacity generalised to transition systems. *International Journal of Information Security* 7, 6 (2008), 421–435.

[14] Michael R Clarkson, Bernd Finkbeiner, Masoud Koleini, Kristopher K Micinski, Markus N Rabe, and César Sánchez. 2014. Temporal logics for hyperproperties. In *International Conference on Principles of Security and Trust*. Springer, 265–284.

[15] Michael R Clarkson and Fred B Schneider. 2010. Hyperproperties. *Journal of Computer Security* 18, 6 (2010), 1157–1210.

[16] Alexandre Donzé. 2010. Breach, a toolbox for verification and parameter synthesis of hybrid systems. In *Computer Aided Verification*. Springer, 167–170.

[17] Alexandre Donzé, Thomas Ferrere, and Oded Maler. 2013. Efficient robust monitoring for STL. In *Computer Aided Verification*. Springer, 264–279.

[18] Georgios E Fainekos, Antoine Girard, and George J Pappas. 2006. Temporal logic verification using simulation. In *International Conference on Formal Modeling and Analysis of Timed Systems*. Springer, 171–186.

[19] Thoshitha T Gamage, Bruce M McMillin, and Thomas P Roth. 2010. Enforcing information flow security properties in cyber-physical systems: A generalized framework based on compensation. In *Computer Software and Applications Conference Workshops (COMPSACW), 2010 IEEE 34th Annual*. IEEE, 158–163.

[20] Thoshitha T. Gamage, Bruce M. McMillin, and Thomas P. Roth. 2010. Enforcing Information Flow Security Properties in Cyber-Physical Systems: A Generalized Framework Based on Compensation.. In *COMPSAC Workshops*. IEEE Computer Society, 158–163.

[21] Antoine Girard and George J Pappas. 2006. Verification using simulation. In *International Workshop on Hybrid Systems: Computation and Control*. Springer, 272–286.

[22] Joseph A Goguen and José Meseguer. 1982. Security policies and security models. In *Security and Privacy, 1982 IEEE Symposium on*. IEEE, 11–11.

[23] Thomas A. Henzinger, Jan Otop, and Roopsha Samanta. 2016. *Lipschitz Robustness of Timed I/O Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 250–267.

[24] Bardh Hoxha, Adel Dokhanchi, and Georgios Fainekos. [n. d.]. Mining parametric temporal logic properties in model-based design for cyber-physical systems. *International Journal on Software Tools for Technology Transfer* ([n. d.]), 1–15.

[25] Xiaoqing Jin, Alexandre Donzé, Jyotirmoy V Deshmukh, and Sanjit A Seshia. 2015. Mining requirements from closed-loop control models. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 34, 11 (2015), 1704–1717.

[26] John Kelsey, Bruce Schneier, David Wagner, and Chris Hall. 1998. Side channel cryptanalysis of product ciphers. In *European Symposium on Research in Computer Security*. Springer, 97–110.

[27] Paul Kocher, Joshua Jaffe, and Benjamin Jun. 1999. Differential power analysis. In *Annual International Cryptology Conference*. Springer, 388–397.

[28] Paul Kocher, Ruby Lee, Gary McGraw, Anand Raghunathan, and Srivaths Moderator-Ravi. 2004. Security as a new dimension in embedded system design. In *Proceedings of the 41st annual Design Automation Conference*. ACM, 753–760.

[29] Paul C Kocher. 1996. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In *Annual International Cryptology Conference*. Springer, 104–113.

[30] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, et al. 2010. Experimental security analysis of a modern automobile. In *Security and Privacy (SP), 2010 IEEE Symposium on*. IEEE, 447–462.

[31] Chunxiao Li, Anand Raghunathan, and Niraj K Jha. 2011. Hijacking an insulin pump: Security attacks and defenses for a diabetes therapy system. In *e-Health Networking Applications and Services (Healthcom), 2011 13th IEEE International Conference on*. IEEE, 150–156.

[32] Lanchao Liu, Mohammad Esmalifalak, Qifeng Ding, Valentine A Emesih, and Zhu Han. 2014. Detecting false data injection attacks on power grid by sparse optimization. *IEEE Transactions on Smart Grid* 5, 2 (2014), 612–621.

[33] Oded Maler and Dejan Nickovic. 2004. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*. Springer, 152–166.

[34] Kebina Manandhar, Xiaojun Cao, Fei Hu, and Yao Liu. 2014. Detection of faults and attacks including false data injection attack in smart grid using kalman filter. *IEEE transactions on control of network systems* 1, 4 (2014), 370–379.

[35] Daryl McCullough. 1987. Specifications for multi-level security and a hook-up. In *Security and Privacy, 1987 IEEE Symposium on*. IEEE, 161–161.

[36] John McLean. 1990. Security models and information flow. In *Research in Security and Privacy, 1990. Proceedings., 1990 IEEE Computer Society Symposium on*. IEEE, 180–187.

[37] John McLean. 1994. A general theory of composition for trace sets closed under selective interleaving functions. In *Research in Security and Privacy, 1994. Proceedings., 1994 IEEE Computer Society Symposium on*. IEEE, 79–93.

[38] Shaunak Mishra, Yasser Shoukry, Nikhil Karamchandani, Suhas Diggavi, and Paulo Tabuada. 2015. Secure state estimation: optimal guarantees against sensor attacks in the presence of noise. In *Information Theory (ISIT), 2015 IEEE International Symposium on*. IEEE, 2929–2933.

[39] Yilin Mo, Tiffany Hyun-Jin Kim, Kenneth Brancik, Dona Dickinson, Heejo Lee, Adrian Perrig, and Bruno Sinopoli. 2012. Cyber–physical security of a smart grid infrastructure. *Proc. IEEE* 100, 1 (2012), 195–209.

[40] Truong Nghiem, Sriram Sankaranarayanan, Georgios Fainekos, Franjo Ivancić, Aarti Gupta, and George J. Pappas. 2010. Monte-carlo Techniques for Falsification of Temporal Properties of Non-linear Hybrid Systems. In *Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control (HSCC '10)*. ACM, New York, NY, USA, 211–220.

[41] Markus N Rabe. 2016. A temporal logic approach to information-flow control. (2016).

[42] Srivaths Ravi, Anand Raghunathan, and Srimat Chakradhar. 2004. Tamper resistance mechanisms for secure embedded systems. In *VLSI Design, 2004. Proceedings. 17th International Conference on*. IEEE, 605–611.

[43] Pankaj Rohatgi. 2009. Electromagnetic attacks and countermeasures. In *Cryptographic Engineering*. Springer, 407–430.

[44] Andrei Sabelfeld and David Sands. 2005. Dimensions and principles of declassification. In *Computer Security Foundations, 2005. CSFW-18 2005. 18th IEEE Workshop*. IEEE, 255–269.

[45] Florian Sagstetter, Martin Lukasiewycz, Sebastian Steinhorst, Marko Wolf, Alexandre Bouard, William R Harris, Somesh Jha, Thomas Peyrin, Axel Poschmann, and Samarjit Chakraborty. 2013. Security challenges in automotive hardware/software architecture design. In *Proceedings of the Conference on Design, Automation and Test in Europe*. EDA Consortium, 458–463.

[46] Yasser Shoukry, Paul Martin, Paulo Tabuada, and Mani Srivastava. 2013. Noninvasive Spoofing Attacks for Anti-lock Braking Systems. In *Proceedings of the 15th International Conference on Cryptographic Hardware and Embedded Systems (CHES'13)*. Springer-Verlag, Berlin, Heidelberg, 55–72.

[47] Geoffrey Smith. 2009. On the foundations of quantitative information flow. In *International Conference on Foundations of Software Science and Computational Structures*. Springer, 288–302.

[48] Jiang Wan, Arquimedes Canedo, and Mohammad Abdullah Al Faruque. 2015. Security-aware functional modeling of cyber-physical systems. In *Emerging Technologies & Factory Automation (ETFA), 2015 IEEE 20th Conference on*. IEEE, 1–4.

[49] J Todd Wittbold and Dale M Johnson. 1990. Information flow in nondeterministic systems. In *Research in Security and Privacy, 1990. Proceedings., 1990 IEEE Computer Society Symposium on*. IEEE, 144–161.

[50] Zhe Xu and A Agung Julius. 2016. Census signal temporal logic inference for multiagent group behavior analysis. *IEEE Transactions on Automation Science and Engineering* (2016).

[51] Steve Zdancewic and Andrew C Myers. 2003. Observational determinism for concurrent program security. In *Computer Security Foundations Workshop, 2003. Proceedings. 16th IEEE*. IEEE, 29–43.

[52] Liang Zou, Naijun Zhan, Shuling Wang, and Martin Fränzle. 2015. Formal Verification of Simulink/Stateflow Diagrams. In *Automated Technology for Verification and Analysis - 13th International Symposium, Shanghai, China*. 464–481.

Cyber-Physical Specification Mismatches

LUAN V. NGUYEN, University of Texas at Arlington
KHAZA ANUARUL HOQUE, University of Oxford
STANLEY BAK, Air Force Research Laboratory
STEVEN DRAGER, Air Force Research Laboratory
TAYLOR T. JOHNSON, Vanderbilt University

Embedded systems use increasingly complex software and are evolving into cyber-physical systems (CPS) with sophisticated interaction and coupling between physical and computational processes. Many CPS operate in safety-critical environments and have stringent certification, reliability, and correctness requirements. These systems undergo changes throughout their lifetimes, where either the software or physical hardware is updated in subsequent design iterations. One source of failure in safety-critical CPS is when there are unstated assumptions in either the physical or cyber parts of the system, and new components do not match those assumptions. In this work, we present an automated method towards identifying unstated assumptions in CPS. Dynamic specifications in the form of candidate invariants of both the software and physical components are identified using dynamic analysis (executing and/or simulating the system implementation or model thereof). A prototype tool called Hynger (for HYbrid iNvariant GEneratoR) was developed that instruments Simulink/Stateflow (SLSF) model diagrams to generate traces in the input format compatible with the Daikon invariant inference tool, which has been extensively applied to software systems. Hynger, in conjunction with Daikon, is able to detect candidate invariants of several CPS case studies. We use the running example of a DC-to-DC power converter, and demonstrate that Hynger can detect a specification mismatch where a tolerance assumed by the software is violated due to a plant change. Another case study of an automotive control system is also introduced to illustrate the power of Hynger and Daikon in automatically identifying cyber-physical specification mismatches.

CCS Concepts: •**Theory of computation →Program specifications;** •**Software and its engineering** →Dynamic analysis;

Additional Key Words and Phrases: Cyber-physical systems, dynamic analysis, specifications

ACM Reference format:
Luan V. Nguyen, Khaza Anuarul Hoque, Stanley Bak, Steven Drager, and Taylor T. Johnson. 201X. Cyber-Physical Specification Mismatches. 1, 1, Article 1 (January 201X), 25 pages.
DOI: 0000001.0000001

Cyber-Physical Specification Mismatches

1 INTRODUCTION

Systems interacting with their physical environments are becoming increasingly dependent upon computers and software, such as in emerging cyber-physical systems (CPS). For instance, typical modern cars utilize hundreds of microprocessors, many communications buses, and a complex interconnection between sensors, actuators, and processors. In the design and development process for

DOI: 0000001.0000001

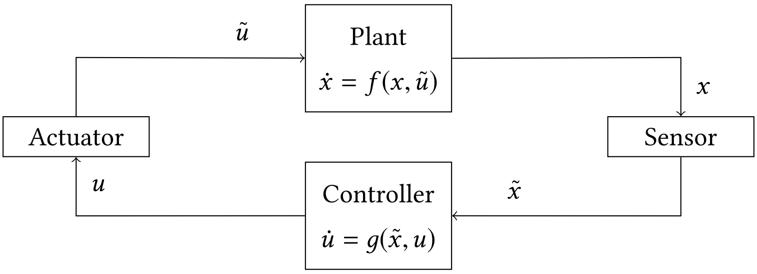


Fig. 1. High-level diagram of a closed-loop control system.

most engineered systems, the vast majority of resources are devoted to ensuring systems meet their specifications [7]. However, in spite of significant technical advances for designing verification and validation such as model checking, Software/Hardware-In-The-Loop (SIL/HIL) testing, automatic test case generation for software, and sophisticated simulators, there still remain products recalled across industries for safety concerns due to software problems and system integration between the cyber and physical subcomponents. The verification community typically focuses on the *developmental verification*, where a model of a system is developed and properties (specifications) are (manually, semi-automatically, or automatically) checked for that system. However, many product recalls and safety disasters induced by software bugs are not a result of design errors, but are the result of either: (*a*) implementation errors, or (*b*) reuse, upgrade, and maintenance errors. Initiatives like a priori Model-Based Design (MBD) are important research efforts and may someday lead to synthesizing provably correct implementations from specifications. However, most systems being designed today still utilize a development process that has engineers writing the software, and systems are integrated with numerous components in a potentially error-prone process. For instance, a typical CPS that has been used widely in control systems is a closed-loop feedback controller shown in Figure 1, where a plant describes physical changes of the environment and a controller represents cyber/software computations corresponding to these changes. The physical evolution of the plant can be sensed and sampled by a sensor, and then fed into the controller. Based on the measurement of the plant provided by the sensor, the controller provides a corresponding control signal to regulate the physical changes in the plant. This control signal is converted by an actuator before sending it to the plant. Such a system may contain different possibilities of failure due to the following main reasons: (*a*) the controller may make wrong assumptions about the plant, sensor or actuator. For example, changing parameters of the plant, sensor, or actuator without updating the controller may produce potential specification mismatches. This controller-reuse issue can lead to safety failures such as the Honda vehicles recalls or the Ariane 5 flight 501 disaster described in Section 2. (*b*) The plant may be influenced by uncontrolled factors (disturbances) from the environment, (*c*) the controller is initially encoded based on wrong information about the physical plant, (*d*) the sensor and actuator may have conversion errors, and (*e*) the control conflicts may arise when using a shared sensor and actuator network.

In this paper, we develop a method to address such challenges that arise in the product evolution and upgrade process in CPS. Our proposed method enables dynamic analysis using well-established software engineering tools for large classes of Simulink/Stateflow (SLSF) models that are frequently used in CPS engineering. In particular, the method infers candidate invariants of SLSF models. Invariants are properties of a system that should always hold, while conditional invariants may hold at certain program points, for example, at the beginning or end of a function call (pre/post conditions). This is important because such models are amenable to formal verification using existing research tools and hybrid system model checkers. Finding invariants can aid this process

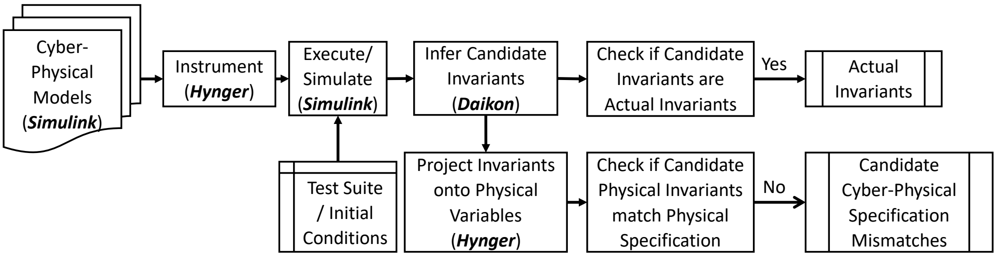

Cyber-Physical Models (Simulink)
Instrument (Hynger)
Execute/ Simulate (Simulink)
Test Suite / Initial Conditions
Infer Candidate Invariants (Daikon)
Project Invariants onto Physical Variables (Hynger)
Check if Candidate Invariants are Actual Invariants
Yes
Actual Invariants
Check if Candidate Physical Invariants match Physical Specification
No
Candidate Cyber-Physical Specification Mismatches

Fig. 2. Preliminary overview of the proposed methodology using Hynger and Daikon to infer candidate invariants and detect specification mismatches.

as they represent potential abstractions with a possibly less complex representation than the set of reachable states. The SLSF block diagrams may be black box components, white box components, or even system implementations (such as when SLSF is used in SIL/HIL simulation). In the case when the underlying SLSF models are known, they may be formalized using hybrid automata [31]. Candidate invariants inferred with our Hynger (for HYbrid iNvariant GEneratoR) software tool in conjunction with Daikon [17, 18] may be formally checked as actual invariants using a hybrid system model checker [20]. Figure 2 shows a preliminary overview of our proposed methodology. As a prelude, we just intuitively demonstrate how Hynger and Daikon can be used to detect specification mismatches. The proposed framework will be fully presented in Section 5.

Contributions. The primary contributions of this paper are: *(a)* the formalization of the cyber-physical specification mismatch problem, *(b)* a methodology for performing template-based automated invariant inference of white box (known) and black box (unknown) CPS models using dynamic analysis, *(c)* the Hynger software tool, which supports instrumenting large classes of SLSF diagrams for dynamic analysis using tools like Daikon, *(d)* a methodology for checking if the inferred invariants are actual invariants by using formal models of the underlying SLSF model diagrams and hybrid systems model checkers such as SpaceEx [20], etc., *(e)* two proof-of-concept CPS case studies using Hynger to automatically identify cyber-physical specification mismatches. These results can be used to help bridging the worlds of actual embedded systems software (e.g., detailed SLSF diagrams and generated C code) with hybrid system models.

Overall, this journal has been substantially extended from our previous work [25]. In fact, we added the formal definitions of cyber-physical specification mismatches, cyber-physical input-output automata, and invariant checking problem to identify whether the inferred invariants are actual invariants. Moreover, two proof-of-concept CPS case studies including a buck converter and an abstract fuel control system are presented to show the capability of Hynger tool in automatically identifying potential cyber-physical specification mismatches of CPSs. The experimental results illustrate the feasibility of using dynamic invariant inference for analysis of embedded and cyber-physical systems. Before presenting the details of our approach, we first illustrate the pitfalls of CPS design reuse by citing examples of critical mistakes in existing, certified systems.

2 CYBER-PHYSICAL DESIGN REUSE AND UPGRADE

In this section, we review cases where CPS design reuse and upgrade have led to failures in existing systems. This motivates the need for our proposed method and our Hynger tool, which can be used to find and formalize unstated assumptions in CPS.

A recent example of a design-reuse problem is the National Highway Transportation and Safety Administration (NHTSA) recall of 1.5 million Honda vehicles (including one of the author's) due to Electronic Control Module (ECM) software problems that could damage the car's transmission, resulting in possible stalls. The root cause of the safety defect was the result of a physical component (a bearing in the transmission) being upgraded to an improved design between different model-year vehicles without appropriate ECM software updates [38]. This problem was widespread because there was a five year delay before the problem was identified, and it was used across model makes and years (e.g., from 2005 – 2010 model year Accords, 2007 – 2010 CR-Vs, and 2005 – 2008 Elements). This difficulty in root-cause analysis emphasizes the point that such problems are probably underreported, and the reuse of components in CPS can lead to widespread serious problems.

Similar design-reuse problems have famously occurred in aviation—the Ariane 5 flight 501 disaster was a result of reusing Ariane 4's software without appropriate updates for the increased thrust of the new rocket [1, 29]. Here, software developers made an assumption about the physical dynamics of the rocket, but the software was reused from Ariane 4, while Ariane 5 had greater thrust, so this assumption was invalid. Finally, when considering the future of CPS, the Defense Advanced Research Projects Agency's System of Systems Integration Technology and Experimentation (DARPA SoSITE) program [32] describes modularized military aviation systems which are capable of rapid component swapping and upgrade. Left unaddressed, issues related to unstated assumptions in components are likely to get worse in future CPS, where changes can occur in the software and hardware.

Besides design-reuse problems, software upgrades without being thoroughly tested and validated may result in an epic system failure. One famous example of this type of problem is the disaster of Mars Climate Orbiter (MCO), developed by NASA's Jet Propulsion Laboratory (JPL). The root-cause of this disaster was that different parts of the software developer team were using different units of measurements. In fact, one part of the ground software upgraded by Lockheed Martin Astronautics (LMA) measured the thrusters in English units of pounds (force)-seconds instead of metric units of Newton-seconds as defined in its original Software Interface Specification (SIS) used by JPL [28, 51]. Therefore, the trajectory of the MCO was erroneously calculated by ground support system computers using the incorrect thruster performance data. This type of software failure occurred due to the lack of adequate communication between different parts of the software team and the uncovered issues of verification and validation processes [51].

2.1 Related Work

The idea evaluated in this work, that of inferring physical system specifications from embedded software in conjunction with physical system models and evaluating them for mismatches, was inspired by previous work finding program specifications for pure software systems [46]. Cyber-physical specification mismatch is closely related to model inconsistency [48], architectural mismatch [21], and requirements consistency [53]. There are many benefits of dynamic analysis such as using implementations instead of models [17, 18, 46] to find dynamic program specifications [46], providing documentation over program evolution and checking if specifications change drastically over program evolution, etc. For one, models are not actually required for analysis, and implementations may be used [17, 18]. The benefit of executing a system implementation is that there are no mismatches between a model (potentially documentation-based) and implementation, since it is not necessary to have a model at all. The candidate specification generated may be viewed as a form of input-output abstraction of the actual implementation. The limitation includes results that are unsound without additional reasoning.

Recently, Medhat and his collaborators introduced a new framework for inferring hybrid automata from black-box implementations of embedded control systems by mining their input/output traces [33]. In their work, the input/output training traces collected from executing a system are clustered and then translated to event sequences. Under several assumptions, hybrid automata representing the behaviors of the system can be inferred using the input/output correlation. Although the work suffers some limitations, their proposed approach is a proof-of-concept of using dynamic analysis to infer the specifications of black-box systems. This work is highly relevant to our proposed method as there is an analogy between inferring hybrid automata and finding a candidate invariant for a black-box system. In fact, both of them can be considered as doing specification inference using dynamic analysis.

There are also several tools such as DepSys [37] and EyePhy [36] that used both static and dynamic analysis to detect and address the control conflict due to dependencies when using multiple CPS applications. Particularly, DepSys is a utility sensing and actuation infrastructure for a smart home that can simultaneously operate multiple CPS applications. The main novelty of DepSys is that it provides a comprehensive strategy to specify, detect and automatically address the control conflicts between sensors and actuators used in a home setting. Similarly, EyePhy is an integrated system that can detect dependencies and then perform a dependency comprehensive analysis across HIL CPS medical applications. A built-in simulator, HumMod, in EyePhy is able to model the complex interactions of the human body using more than 7,800 physiological variables. HumMod demonstrates the model parameters and the quantitative relationship among them in XML files that makes it easier to update the physiological models without the recompilation of the whole system. EyePhy can be considered as the first tool that performs the dependency analysis across applications' control actions on the human body. Additionally, the sensor networks with devices used in smart homes or medical devices can be built using the family of Smart Transducer Interface Standards (IEEE 1451). IEEE 1451 has been developed in order to provide the common communication interfaces for connecting transducers (sensors or actuators) to their instrumentation systems or control networks [27]. The Transducer Electronic Data Sheets (TEDS) embedded in smart transducers are memory devices, which store the manufacture-related information of the transducer such as manufacture ID, measurement ranges, serial number, etc. Thus, TEDS allows transducers to be self-identified and self-descriptive to the device networks. It also provides a standardized mechanism to facilitate the plug and play of transducers with different control networks. Hence, IEEE 1451 enables the access of transducer data through a common set of interfaces, allowing users to select transducers and networks for their applications. This advantage is crucial in facilitating the device and data interoperability, detecting and resolving conflicts due to dependencies when concurrently using multiple transducers in the device networks.

Finding specifications is a maturing field within software engineering [10, 11, 17, 18, 46]. Daikon, which is used by Hynger, processes program traces to generate invariants [17, 18]. For several languages (C, C++, etc.), this process is performed without access to the source code by instrumenting the compiled program using Valgrind [39]. This makes it difficult to use on non-x86/x86-64 platforms (although Valgrind is gaining access to other architectures), which is a serious limitation, as most embedded platforms utilize other architectures (e.g., ARM, AVR, PIC, 8051, MSP430, etc.). Due in part to these limitations, Hynger instruments architecture-independent SLSF diagrams directly. In the long run, the Hynger tool is envisioned to take an arbitrary SLSF model, instrument it, then analyze the resulting traces with dynamic analysis to identify broad classes of cyber-physical specification mismatches.

The most closely related work using Daikon is to find candidate invariants of hybrid models of biological system [9], and this also illustrates a proof-of-concept of using Daikon as a trace analyzer for non-purely software systems. Daikon can generate invariants of many forms for variables and data structures, such as: ranges ($a \leq x \leq b$), linear ($y = ax + b$), variable ordering ($x \leq y$), sortedness of lists, etc. Daikon works by instrumenting source code and/or compiled binaries with changes that allow for looking at variable values, then Daikon essentially checks if variables satisfy some template invariants. For instance, if an integer variable x is observed to always be smaller than some number, say 50, Daikon may generate a candidate invariant of $x \leq 50$. Based on many advantages of using Daikon as a trace analyzer [17, 18], we prefer to use Hynger with Daikon to infer candidate invariants in our proposed framework. However, we note that Hynger can generate a trace file in many input formats that are compatible with other invariant-inference tools using dynamic analysis not just Daikon. Other research tools like DySy [11] and commercial tools like Agitagor [10] can be used for generating candidate invariants for other languages.

3 CYBER-PHYSICAL SYSTEM MODELS

The approach presented in this paper applies to the systems with formal models, informal models, and unknown models/implementations. The primary assumption is that interfaces to the models or systems are available as SLSF blocks. There are two main categories of blocks appearing in an SLSF diagram that are supported by our method, white box and black box systems. The white box systems may contain: (*a*) known, informal models, (*b*) known, informal implementations, or (*c*) known, formal models (e.g., hybrid automata, or more precisely, classes of SLSF diagrams that may be converted to hybrid automata [31]). The black box systems may be completely unknown, and may contain: (*a*) unknown implementations (e.g., compiled executable binaries with no source available, such as commercial off-the-shelf [COTS] components and other third-party systems), (*b*) unknown models, and (*c*) actual cyber-physical systems (e.g., embedded controllers, networked computers, and physical plants, all that may show up in HIL/SIL simulations interfaced with SLSF).

Next, we define a structure of CPS models used in SLSF. We will not define formal semantics of this structure or SLSF diagrams in this paper. However, in the case where an SLSF diagram is a white box and formal semantics may be defined, a formal framework like hybrid input/output automata (HIOA) [30] may be used to specify the semantics, such as done in the HyLink tool [31]. Additionally, if an SLSF diagram is a white box and linear, we may also be able to use SL2SX Translator for transforming it into a corresponding formal model [34]. Interested readers can find some graphical examples of the translation in [31, 34]. Other formalisms like actors and hierarchical state machines are commonly used for formal modeling of other diagrammatic frameworks similar to SLSF [2, 8, 52, 54]. Given a formal model $\mathcal{A}$ and candidate specification Σ (e.g., found using Hynger), we can check if $\mathcal{A}$ meets the specification, i.e., $\mathcal{A} \models \Sigma$ by using a hybrid system model checker like SpaceEx [20]. In some instances, we know when an SLSF diagram corresponds precisely to a hybrid automaton model [31], and in these cases, we can check if candidate invariants found with Hynger are actual invariants.

First, we define the hierarchy represented by SLSF diagrams.

Definition 3.1 (SLSF diagram). An SLSF diagram is a tuple $\mathcal{A} \triangleq \langle M, E, \mathsf{V} \rangle$, where:

- M is a set of blocks (vertices) that represent block diagrams (and sub-blocks/models),
- $E \subseteq M \times M$ is a set of edges between blocks representing a parent-child hierarchy, and
- V is a set of variables, written as $\mathsf{V} \triangleq \bigcup_{v \in M} \mathsf{V}(v)$, where $\mathsf{V}(v)$ is a set of variables for each block $v \in M$.

According to Definition 3.1, the graph $G \triangleq (M, E)$ defined by the vertices (blocks) M and edges E is a rooted tree, where M are block diagrams and E represents a parent-child hierarchical relationship (e.g., sub-modules and sub-blocks). Here, the root (i.e., top-level) block diagram of the model is the unique root of the tree, which we denote as $root(M)$. For a block $v \in M$, the *children* of v are denoted as $children(v)$ and defined as the set of blocks $\{w \in M \mid w \in E(v)\}$. For a block $v \in M$, the *parent* of v is denoted as $parent(v)$ and is defined as the singleton set $\{w \in M \mid v \in children(w)\}$.

Clearly, $parent(root(M)) = \emptyset$. For a block $v \in M$, the *ancestors* of v are denoted as $ancestors(v)$ and defined inductively as the set of blocks $\{w \in M \mid v \ \cup \ w \in children(v) \ \cup \ children(w)\}$ (or equivalently, as the transitive closure of $children(v)$).

For a block $v \in M$, the set of variables of v is $\mathsf{V}(v)$ and is partitioned into sets of input and output variables, written respectively as $\mathsf{V}_I(v)$ and $\mathsf{V}_O(v)$, and we have $\mathsf{V}(v) = \mathsf{V}_I(v) \cup \mathsf{V}_O(v)$. A *variable* $x \in \mathsf{V}(v)$ is a name for referring to some state of $\mathcal{A}$, and is associated with a data type denoted $type(x)$. Typical data types are reals, floating points, arrays, lists, etc. The valuation of a variable $x \in \mathsf{V}(v)$ is the set of all values it may take and is denoted $val(x)$. The state-space of $\mathcal{A}$ is the set of valuations of all the variables V. An element s of the state-space is called a state, and a trace is a sequence of states. The SLSF diagram may also have internal (local) variables, although they are not externally visible, so we do not include them, as only input/output interfaces are visible for external observation and instrumentation.

Next, we define CPS models that appear in SLSF diagrams.

Definition 3.2 (CPS model). A CPS model is an SLSF diagram with a set of n typed variables, $\mathsf{V} = \{x_1, x_2, \ldots, x_n\}$, which is classified into two subsets as follows.

- $\mathsf{V}_P = \{\alpha_1, \alpha_2, \ldots, \alpha_{n_p}\}$ is a set of $n_p \leq n$ physical variables such that their values are continuously updated, and
- $\mathsf{V}_C = \{\beta_1, \beta_2, \ldots, \beta_{n_c}\}$ is a set of n_c cyber variables that are discretely updated, where $n = n_p + n_c$.

Here, the set of variables for each block of a CPS model is also partitioned into sets of physical and cyber variables, $\mathsf{V}(v) = \mathsf{V}_P(v) \cup \mathsf{V}_C(v)$. In practice, this may be accomplished with subtyping using, for example, an overloaded type for floats or fixed points used for approximations of real variables (e.g., in C, `typedef double physical; typedef physical temperature;`). The dynamic changes of the variables of the CPS model may be described using different SLSF blocks such as S-Function block, look-up table, etc. In case the CPS model is a white-box and simple enough, we may translate it to a formal framework like HIOA (e.g using Hylink). In fact, we can specify a set of real-valued variables and their dynamic changes for the converted formal model based on capturing the output variables from unit delay, integrator, state-space blocks in the corresponding SLSF diagram [3]. Moreover, we note that the input and output variables are disjoint, and the cyber and physical variables are disjoint, although these are not all mutually disjoint. Hence, we further classify the set of variables $\mathsf{V}(v)$ into different types as follows.

Definition 3.3 (Variable Classification). For a block $v \in M$, a variable $x \in \mathsf{V}(v)$ is considered as:

- an *input cyber variable* if $x \in \mathsf{V}_C(v)$ and $x \in \mathsf{V}_I(v)$,
- an *output cyber variable* if $x \in \mathsf{V}_C(v)$ and $x \in \mathsf{V}_O(v)$,
- an *input physical variable* if $x \in \mathsf{V}_P(v)$ and $x \in \mathsf{V}_I(v)$, or
- an *output physical variable* if $x \in \mathsf{V}_P(v)$ and $x \in \mathsf{V}_O(v)$.

We extend these notations in Definition 3.3 naturally to sets of variables if *all* variables in a set of variables fall into these classes, and will reference them as such. An arbitrary set of variables may not be mutually disjoint from each of the input, output, cyber, and physical variables. Thus, for a set of variables $X \subseteq \mathsf{V}$, we say: (*a*) X is *cyber-physical* if there exist both cyber and physical variables in X, (*b*) X is *input-output* if there exist both input and output variables in X, and (*c*) X is *cyber input-output*, *physical input-output*, *cyber-physical input*, or *cyber-physical output* for the other natural permutations.

Next, using these variable classes, we define classes of SLSF blocks appearing in SLSF diagrams. For a block $v \in M$, we say: (*a*) v is a *cyber-physical* block if there exist both cyber and physical variables in $\mathsf{V}(v)$, (b) v is a *cyber* block if there exist *only* cyber variables in $\mathsf{V}(v)$, and (c) v is a *physical* block if there exist *only* physical variables in $\mathsf{V}(v)$.

Cyber-Physical Variable Interactions. Next, we will formalize a notion of influence between cyber and physical models and their variables. For example, consider a typical closed-loop plant-controller architecture, where outputs of a plant are sensed, used as inputs to a controller, and outputs of the controller are converted by actuators as inputs to the plant (and potentially disturbances affect everything). Generally, we would say the plant is a physical model, the controller is a cyber model, and the sensors and actuators are cyber-physical models. However, it is clear that the physical variables of the plant affect the cyber variables of the controller, and vice-versa, albeit not directly, but through the transitive closure of input-output connections over all blocks in the SLSF diagram. We note that this is related to the notion of tainted variables in program analysis that is popular in security [49]. To formalize this notion, we specify interconnections between input and output variables between blocks $v \in M$ at the same hierarchical level in the diagram.

Input-output connections may only exist between models with the same parent (i.e., those in the same hierarchical structure). For a block $v \in M$, we denote all blocks with the same parent as $siblings(v)$, which is defined as the set $\{w \in M \mid parent(w) = parent(v)\}$. Output variables of a block $v \in M$ may be connected to input variables of a block $w \in M$. Let $G_{\mathsf{V}} \overset{\Delta}{=} (V_{\mathsf{V}}, E_{\mathsf{V}})$ be a directed graph where the vertices V_{V} are variables of blocks $v \in M$ and the edges specify the interconnection between output variables to input variables for some model $w \in siblings(v)$, and we have $E_{\mathsf{V}} \subseteq \mathsf{V}(v) \times \mathsf{V}(w)$. In general, for a fixed block $v \in M$ and variable $x \in \mathsf{V}(v)$, this interconnection relation is a tree, rooted at the output variable x and connected to possibly many input variables of other blocks $w \in M$ for $w \neq v$. For two blocks $v, w \in M$, we say v *connects to* w if there exists an output variable $y \in \mathsf{V}_O(v)$ and an input variable $u \in \mathsf{V}_I(w)$ with $E_{\mathsf{V}}(u) = y$, denoted $v \hookrightarrow w$. For two blocks $v, w \in M$, we say v *has a path to* w if w is in the transitive closure of blocks that v connects to (i.e., $v \hookrightarrow^{*} w$), denoted $v \rightsquigarrow w$. We note that the $\rightsquigarrow$ relation may have cycles, and such cases arise in feedback control loops. For a block $v \in M$, for an input variable $u \in \mathsf{V}_I(v)$ and output variable $y \in \mathsf{V}_O(v)$, we say u *directly influences* y if the value of y changes as a function of u.[1] Finally, for two blocks $v, w \in M$ such that $v \rightsquigarrow w$, for an output variable $y \in \mathsf{V}_O(v)$ and an input variable $u \in \mathsf{V}_I(w)$, we say y *influences* u if there exists a sequence of directly influenced variables between y and u. Thus, we can see that a cyber variable in one model may influence a physical variable in another model (or even its own model if there is a cycle), and vice-versa. The *software physical variables* are all cyber variables that are influenced by physical variables, and are denoted V_{SP}. Typical examples of software physical variables include those used for sensed and sampled measurements, variables used in feedback control calculations, etc.

Example 3.4. Here, we describe a CPS case study used throughout the remainder of the paper for illustrating concepts. The case study is a DC-to-DC power converter (like buck, boost, and buck-boost converters) [40], all of which have similar modeling, but we focus particularly on a buck converter. The buck converter has two real-valued state variables modeling the inductor current i_L and the capacitor voltage V_C. These state variables are written in vector form as: $x = [i_L; V_C]$. The dynamics of the continuous variables in each mode $m \in \{Open, Close, DCM\}$ are specified as linear (affine) differential equations: $\dot{x} = A_m x + B_m u$, where $u = V_S$ is a source voltage. The A_m matrices consist of $L > 0$, $R > 0$, $C > 0$ real-valued constants, respectively representing inductance (in Henries), resistance (in Ohms), and capacitance (in Farads). A buck converter takes an input voltage of say 5V and "bucks" or drops the voltage to some lower DC voltage, say 2.5V. These circuits

[1] Internally the blocks may be very sophisticated, could represent complex physical systems, could be Turing complete, etc., so we use this abstract notion.

are used in many electronic devices (e.g., personal computers, cellphones, embedded systems, aircraft, satellites, cars). These circuits are also used as modular components in a variety of novel power electronics architectures, such as AC/DC microgrids and distributed DC-to-AC multilevel inverters [42].

The general architecture of the buck converter that we focus on consists of a plant (physical system) model and a controller (cyber model/software), along with models of actuators and sensors interfacing the plant and controller. A controller for the buck converter may be constructed as a hysteresis controller, which changes the mode of the buck converter plant based on the measured output voltage [22]. In fact, the converter is meant to transform a given source voltage V_S to create an output voltage V_{out} approximately equal to a desired reference voltage (or set-point) V_{ref}. To accomplish this, the switch controlling whether V_S is connected to the output or not is toggled depending on whether $V_{out} > V_{ref}$ or $V_{out} < V_{ref}$. In practice, to avoid switching too often, a hysteresis band is used and switches occur when $V_{out} > V_{ref} + V_{tol}$ or $V_{out} < V_{ref} - V_{tol}$. The choice of V_{tol}, along with the system dynamics, will determine the voltage ripple V_{rip} about the set-point V_{ref}. Typical specifications require the voltage ripple to be small, so that the output voltage V_{out} is approximately V_{ref}, that is, V_{rip} is chosen so that for $V_{out} = V_{ref} \pm V_{rip}$, we have $V_{out} \approx V_{ref}$. The sensor model performs quantization and sampling, as would occur in typical analog to digital conversion (ADC) used to digitize analog signal measurements. The actuator models likewise perform the inverse process of digital to analog conversion (DAC) to convert the digital (cyber) signals to analog signals.

Generally, we can model the plant as a physical block, the hysteresis controller as a cyber block, and the sensors and actuators as cyber-physical blocks in SLSF. The plant voltage is an output physical variable that affects the output cyber variable—a switching mode signal that enables the transition between each mode in the plant—of the controller, and vice-versa. This interaction between the plant and the controller is accomplished through the transitive closure of input-output connections with the sensor and the actuator in the SLSF model. We will formalize specifications and mismatches of the buck converter in Section 4. As a prelude, we highlight that Hynger finds its candidate invariant (that can be shown to be an actual invariant when modeled as a hybrid automaton [22, 26, 40]).

3.1 Cyber-Physical Input-Output Automata

To further investigate cyber-physical specification mismatches of CPS models, we consider ones that may be formally represented as cyber-physical input-output automata.

Definition 3.5. A cyber-physical input-output automaton (CPIOA) $\tilde{\mathcal{A}}$ is a tuple, $\tilde{\mathcal{A}} \triangleq \langle Loc, Var, Flow, Inv, Traj, Lab, Trans, Init\rangle$, consisting of the following components:

- *Loc*: a finite set of discrete locations.
- *Var*: a finite set of n continuous, real-valued variables, where $\forall x \in Var$, $val(x) \in \mathbb{R}$ and $val(x)$ is a valuation—a function mapping x to a point in its type—here, $\mathbb{R}$; and $Q \triangleq Loc \times \mathbb{R}^n$ is the state space. *Var* is the disjoint of a set of input variables I and a set of output variables O. Furthermore, C and P classify *Var* into sets of cyber and physical variables, respectively.
- *Inv*: a finite set of invariants for each discrete location, $\forall \ell \in Loc$, $Inv(\ell) \subseteq \mathbb{R}^n$.
- *Flow*: a finite set of derivatives for each continuous variable $x \in Var$, and $Flow(\ell, x) \subseteq \mathbb{R}^n$ describes the continuous dynamics of each location $\ell \in Loc$. if x is a physical variable, $Flow(\ell, x)$ is a non-zero Lipschitz continuous differential equation over time. Otherwise, if x is a cyber variable, $Flow(\ell, x) = 0$.
- *Traj*: a finite set of continuous trajectory models the valuations of variables over an interval of real time $[0, T]$. Let Δ_0, Δ_t and Δ_T be the valuations of variable x at time points 0, t, and

T respectively, $\forall t \in [0, T]$, $\forall x \in \mathit{Var}$, $\exists \ell \in \mathit{Loc}$, a trajectory $\tau \in \mathit{Traj}$ is a mapping function $\tau : [0, T] \longrightarrow \mathit{val}(\mathit{Var})$ such that:
  - ⋄ $\Delta_t = \Delta_0 + \int_{\delta=0}^{t} \mathit{Flow}(\ell, x) d\delta$, and
  - ⋄ $\Delta_0 \models \mathit{Inv}(\ell)$, $\Delta_t \models \mathit{Inv}(\ell)$, and $\Delta_T \models \mathit{Inv}(\ell)$.
- *Lab*: a finite set of synchronization labels.
- *Trans*: a finite set of transitions between locations; each transition is a tuple $\gamma \triangleq \langle \ell, \ell', g, u \rangle$, which can be taken from source location ℓ to destination location ℓ' when a guard condition g is satisfied, and the post-state is updated by an update map u.
- *Init* is an initial condition, which consists of a set of locations in *Loc* and a formula over *Var*, so that $\mathit{Init} \subseteq \boldsymbol{Q}$.

Next, we define the semantics of a CPIOA $\tilde{\mathcal{A}}$ in term of executions. An execution of $\tilde{\mathcal{A}}$ is a sequence of states, written as $\rho \triangleq s_0 \longrightarrow s_1 \longrightarrow s_2 \longrightarrow \ldots$, where $s_0 \in \mathit{Init}$, and $s_i \longrightarrow s_{i+1}$ is the update from the current-state s_i to the post-state s_{i+1}, that is specified by the transition relations of the CPIOA $\tilde{\mathcal{A}}$ including: (a) a discrete transition that demonstrates the instantaneous state update, or (b) a continuous trajectory that represents the state update over a real time interval. We say a state s_k is reachable from an initial state s_0 if there exists an execution $\rho \triangleq s_0 \longrightarrow s_1 \longrightarrow \ldots \longrightarrow s_k$.

Invariant Property. An *invariant property* φ of a CPIOA $\tilde{\mathcal{A}}$ is a formula over *Var* and *Loc* that is always true for every reachable state of $\tilde{\mathcal{A}}$. Formally, we say $\tilde{\mathcal{A}} \models \varphi$ iff $\forall s \in \mathit{Reach}(\tilde{\mathcal{A}})$, $s \models \varphi$, where $\mathit{Reach}(\tilde{\mathcal{A}})$ denotes the set of reachable states of $\tilde{\mathcal{A}}$.

Parallel Composition. Consider two CPIOAs $\tilde{\mathcal{A}}_1 \triangleq \langle \mathit{Loc}_1$, Var_1, Inv_1, Flow_1, Traj_1, Lab_1, Trans_1, $\mathit{Init}_1 \rangle$, and $\tilde{\mathcal{A}}_2 \triangleq \langle \mathit{Loc}_2$, Var_2, Inv_2, Flow_2, Traj_2, Lab_2, Trans_2, $\mathit{Init}_2 \rangle$, we consider that $\tilde{\mathcal{A}}_1$ and $\tilde{\mathcal{A}}_2$ is *compatible* if (a) $I_1 \subseteq O_2$, (b) $I_2 \subseteq O_1$, and (c) $O_1 \cap O_2 = \emptyset$. The parallel composition operation combines two compatible CPIOAs into a single CPIOA that represents the synchronous interaction between these two CPIOA when running simultaneously.

Definition 3.6 (Parallel Composition). Given two compatible CPIOAs $\tilde{\mathcal{A}}_1$ and $\tilde{\mathcal{A}}_2$, the parallel composition of $\tilde{\mathcal{A}}_1$ and $\tilde{\mathcal{A}}_2$ is a CPIOA $\tilde{\mathcal{A}}$, written as $\tilde{\mathcal{A}} \triangleq \tilde{\mathcal{A}}_1 \| \tilde{\mathcal{A}}_2$, where:

- $\mathit{Loc} = \mathit{Loc}_1 \times \mathit{Loc}_2$,
- $\mathit{Var} = \mathit{Var}_1 \cup \mathit{Var}_2$,
- $\boldsymbol{Q} = \boldsymbol{Q}_1 \times \boldsymbol{Q}_2$,
- $O = O_1 \cup O_2$,
- $I = (I_1 \cup I_2) \setminus O$,
- $\forall \ell_1, \ell_2 \in \mathit{Loc}$, $\mathit{Inv}(\ell_1, \ell_2) = \mathit{Inv}_1(\ell_1) \wedge \mathit{Inv}_2(\ell_2)$
- $\forall \ell_1, \ell_2 \in \mathit{Loc}$, $\forall x \in \mathit{Var}$, $((\ell_1, \ell_2), \mathit{val}(x) \in \mathit{Init})$ iff $(\ell_1, \mathit{val}(x)) \in \mathit{Init}_1 \wedge (\ell_2, \mathit{val}(x)) \in \mathit{Init}_2$,
- $\mathit{Lab} = \mathit{Lab}_1 \cup \mathit{Lab}_2$,
- $\forall i \in \{1, 2\}$, there is a trajectory $\tau \in \mathit{Traj}$ iff $\tau \downarrow (\mathit{Loc}_i \cup \mathit{Var}_i) \in \mathit{Traj}_i$, where $\tau \downarrow (\mathit{Loc}_i \cup \mathit{Var}_i)$ denotes the projection of τ onto the sets of variables and locations of component i.
- Given $\gamma_1 \in \mathit{Trans}_1$, $\gamma_1 \triangleq \left\langle \ell_1, \ell_1', g_1, u_1 \right\rangle$ and $\gamma_2 \in \mathit{Trans}_2$, $\gamma_2 \triangleq \left\langle \ell_2, \ell_2', g_2, u_2 \right\rangle$, there exists a transition $\gamma \in \mathit{Trans}$, $\gamma \triangleq \langle \ell, \ell', g, u \rangle$ iff:
  - ⋄ $\ell = (\ell_1, \ell_2)$, $\ell' = (\ell_1', \ell_2)$, $g = g_1$, and $u = u_1$, or
  - ⋄ $\ell = (\ell_1, \ell_2)$, $\ell' = (\ell_1, \ell_2')$, $g = g_2$, and $u = u_2$, or
  - ⋄ $\ell = (\ell_1, \ell_2)$, $\ell' = (\ell_1', \ell_2')$, $g = g_1 \wedge g_2$, and $u = u_1 \cup u_2$.

Closed-loop CPIOA. One type of CPS model that we focus on in this paper is a closed-loop model, e.g., the closed-loop buck converter. Such a model can be formally represented as a closed-loop

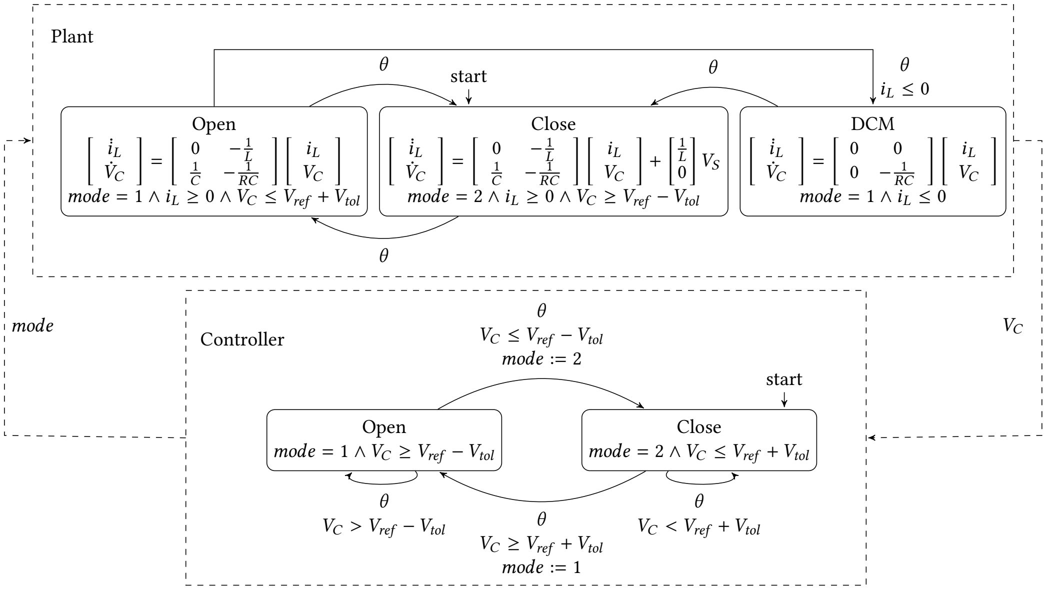

Plant
θ
start
θ
θ
$i_L \leq 0$
Open
$\begin{bmatrix} \dot{i}_L \\ \dot{V}_C \end{bmatrix} = \begin{bmatrix} 0 & -\frac{1}{L} \\ \frac{1}{C} & -\frac{1}{RC} \end{bmatrix} \begin{bmatrix} i_L \\ V_C \end{bmatrix}$
$mode = 1 \wedge i_L \geq 0 \wedge V_C \leq V_{ref} + V_{tol}$
Close
$\begin{bmatrix} \dot{i}_L \\ \dot{V}_C \end{bmatrix} = \begin{bmatrix} 0 & -\frac{1}{L} \\ \frac{1}{C} & -\frac{1}{RC} \end{bmatrix} \begin{bmatrix} i_L \\ V_C \end{bmatrix} + \begin{bmatrix} \frac{1}{L} \\ 0 \end{bmatrix} V_S$
$mode = 2 \wedge i_L \geq 0 \wedge V_C \geq V_{ref} - V_{tol}$
DCM
$\begin{bmatrix} \dot{i}_L \\ \dot{V}_C \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & -\frac{1}{RC} \end{bmatrix} \begin{bmatrix} i_L \\ V_C \end{bmatrix}$
$mode = 1 \wedge i_L \leq 0$
θ
mode
V_C
Controller
θ
$V_C \leq V_{ref} - V_{tol}$
$mode := 2$
start
Open
$mode = 1 \wedge V_C \geq V_{ref} - V_{tol}$
Close
$mode = 2 \wedge V_C \leq V_{ref} + V_{tol}$
θ
$V_C > V_{ref} - V_{tol}$
θ
$V_C \geq V_{ref} + V_{tol}$
$mode := 1$
θ
$V_C < V_{ref} + V_{tol}$

Fig. 3. A hybrid automaton models the buck converter plant with hysteresis controller.

CPIOA, which is a parallel composition of a plant and controller CPIOA. The plant CPIOA has continuous dynamics modeled by ordinary differential equations, and the controller CPIOA can be purely discrete. For instance, the hybrid automaton of the closed-loop buck converter (without sensor and actuator) shown in Figure 3 can be considered as one closed-loop CPIOA, where θ is a synchronization label and *mode* is a discrete control signal. The capacitor voltage variable V_C is not only an output physical variable for the plant CPIOA, but is also an input cyber variable of the controller CPIOA. In this case, we can check whether the candidate invariants of the closed-loop buck converter found with Hynger and Daikon are actual invariants by investigating its formal model (e.g., a closed-loop CPIOA shown in Figure 3) using some hybrid systems model checkers such as SpaceEx [20].

3.2 Candidate Invariant Checking Problem

The formal definition of the candidate invariant checking problem for CPS is described as follows.

Definition 3.7 (Candidate Invariant Checking). Given a CPS model $\mathcal{A}$ with a set of candidate invariants $\hat{\Phi}$, $\tilde{\mathcal{A}}$ is a formal model converted from $\mathcal{A}$, a candidate invariant $\hat{\varphi} \in \hat{\Phi}$ is considered as an actually invariant property of $\tilde{\mathcal{A}}$ iff $Reach(\tilde{\mathcal{A}}) \models \hat{\varphi}$.

According to Definition 3.7, if a CPS model $\mathcal{A}$ is a white box system that can be represented in terms of a formal model such as a CPIOA $\tilde{\mathcal{A}}$, a hybrid system model checker may be used to check whether $\hat{\varphi}$ is an invariant property of $\tilde{\mathcal{A}}$. If there exists any reachable state of $\tilde{\mathcal{A}}$ that does not satisfy $\hat{\varphi}$, we can conclude that $\hat{\varphi}$ is not an actual invariant of the CPS model $\mathcal{A}$.

4 CYBER-PHYSICAL SPECIFICATIONS AND MISMATCHES

In this section, we will formalize the concept of candidate cyber-physical specification mismatches of CPS, and introduce a potential method to detect such specification mismatches.

4.1 Cyber-Physical Specifications

Our goal is to find specifications that are invariants or conditional invariants, so we do not consider more general temporal logic formulas. Under this assumption, a *specification* is equivalent to a predicate over the state-space of the system. Equivalently, a specification is a multi-sorted first-order logic (FOL) sentence (of a restricted class), so we assume the specification may be represented in the Satisfiability Modulo Theories (SMT) library standard language [6, 35]. Under these assumptions, candidate invariants may be specified as quantifier-free SMT formulas over the variables of the SLSF model, where the type of a variable corresponds to the SMT sort. For a formula ϕ, let $vars(\phi)$ be the set of variables appearing in ϕ. For a formula ϕ: (a) if $vars(\phi)$ are all physical, then ϕ is a *physical specification*, (b) if $vars(\phi)$ are all cyber, then ϕ is a *cyber specification*, and (c) if $vars(\phi)$ consists of both cyber and physical variables, then ϕ is a *cyber-physical specification.*

Next, while we will try to infer interesting specifications ϕ using dynamic analysis later in the paper, we first highlight examples of specifications made a priori in system design, as these are necessary to define specification mismatches. Let Σ be a set of specifications for $\mathcal{A}$, which is a set of formulas over the variables of $\mathcal{A}$. Referring to Figure 4, we separate the specification Σ into sets of cyber and physical specifications, written respectively as Σ_C and Σ_P. These specifications include assumptions about the physical environment, such as the value of gravitational force, temperature bounds, time constants, etc. The physical specification also includes assumptions about the physical system's behavior and subcomponents, such as motor torque limits, temperature bounds of components, sampling rates, velocity limits, etc. Here Σ_C denotes the set of cyber specifications. The cyber specifications include assumptions about software-physical interfaces, such as ADC resolution, DAC resolution, sampling rates, etc. It also includes assumptions about the software system, subcomponents, and software-software interfaces, such as data formats, control flow, event orderings, etc. For example, the buck converter has the following physical specifications:

$$\begin{aligned}\sigma_P^1 &\triangleq t \geq t_s \Rightarrow V_{out}(t) = V_{ref}(t) \pm V_{rip},\\ \sigma_P^2 &\triangleq V_S(t) = V_S(0) \pm \delta_S,\\ \sigma_P^3 &\triangleq V_{ref}(t) = V_{ref}(0) \pm \delta_{ref},\end{aligned}$$

and $\Sigma_P \triangleq \{\sigma_P^1, \sigma_P^2, \sigma_P^3\}$. Here, σ_P^1 states that after some amount of constant startup time t_s, the output of the buck converter $V_{out}(t)$ remains near a reference (desired) output voltage $V_{ref}(t)$. Both σ_P^2 and σ_P^3 specify assumptions about the buck converter's environment, namely that its source voltage V_S and reference voltage V_{ref} always remain near their initial values. We note that while time may not typically be thought of as a state of the system, it can be encoded in this way easily, for example, by including a state variable t with $\dot{t} = 1$. To evaluate whether $\mathcal{A}$ has cyber-physical specification mismatches, we hypothesize that the cyber specification contains (at least a subset) of the physical specification. This process is made more explicit in Figure 4 and described next.

4.2 Cyber-Physical Specification Mismatches

A CPS model or implementation will be provided as an SLSF diagram, denoted $\mathcal{A}$ as formalized above. Next, $\mathcal{A}$ is instrumented using the Hynger yielding a modified SLSF diagram $\hat{\mathcal{A}}$. Now, $\hat{\mathcal{A}}$ is executed to generate a set of sampled, finite-precision traces T for each initial condition θ in a set of initial conditions Θ, which effectively corresponds to a test suite. The traces T are analyzed using dynamic analysis methods, such as Daikon, to generate a set of candidate invariants $\hat{\Phi}$, each element $\hat{\varphi}$ of which may be checked as actual invariants if $\mathcal{A}$ corresponds to a formal model (e.g., a CPIOA) or may be converted to one, $\tilde{\mathcal{A}}$. If that is the case, then a hybrid system model checker may be employed to see if $\hat{\varphi}$ is an actual invariant φ, and the set of actual invariants Φ is collected.

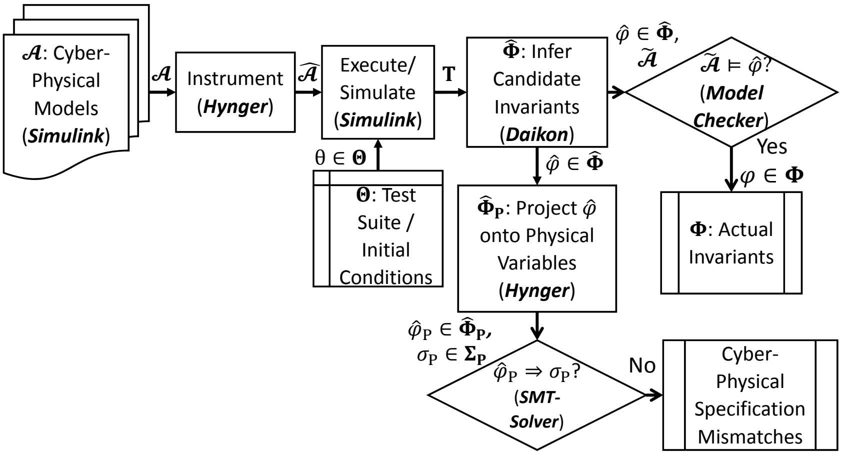

$\mathcal{A}$: Cyber-Physical Models (*Simulink*)
$\mathcal{A}$
Instrument (*Hynger*)
$\widetilde{\mathcal{A}}$
Execute/ Simulate (*Simulink*)
T
$\hat{\Phi}$: Infer Candidate Invariants (*Daikon*)
$\hat{\varphi} \in \hat{\Phi}$, $\widetilde{\mathcal{A}}$
$\widetilde{\mathcal{A}} \models \hat{\varphi}$? (*Model Checker*)
Yes
$\varphi \in \Phi$
Φ: Actual Invariants
$\theta \in \Theta$
Θ: Test Suite / Initial Conditions
$\hat{\varphi} \in \hat{\Phi}$
$\hat{\Phi}_P$: Project $\hat{\varphi}$ onto Physical Variables (*Hynger*)
$\hat{\varphi}_P \in \hat{\Phi}_P$, $\sigma_P \in \Sigma_P$
$\hat{\varphi}_P \Rightarrow \sigma_P$? (*SMT-Solver*)
No
Cyber-Physical Specification Mismatches

Fig. 4. Hynger overview, inference of physical specifications assumed by software, and cyber-physical specification mismatch identification.

Definition 4.1 (Cyber-Physical Specification Mismatch). Given an SLSF diagram $\mathcal{A}$ with a set of actual physical specifications Σ_P, let $\hat{\Phi}_P \triangleq \hat{\Phi} \downarrow V_{SP}$ be a set of candidate physical invariant, $\mathcal{A}$ has a cyber-physical specification mismatch iff: $\exists \sigma_P \in \Sigma_P, \forall \hat{\varphi}_P \in \hat{\Phi}_P, \sigma_P \not\models \hat{\varphi}_P$.

In Definition 4.1, $\hat{\Phi} \downarrow V_{SP}$ denotes the projection or the restriction of $\hat{\Phi}$ to the set of software physical variable V_{SP}. In all cases, each candidate invariant $\hat{\varphi} \in \hat{\Phi}$ is projected (restricted) onto the software physical variables V_{SP} to yield a candidate physical invariant $\hat{\varphi}_P$ and corresponding set $\hat{\Phi}_P$. Such a projection may be computed using quantifier elimination methods available in many modern SMT solvers, such as Z3 [13][2]. Now, $\hat{\Phi}_P$ corresponds to the candidate, inferred physical invariants from the perspective of the cyber-physical system, each element of which may be compared to each element σ_P of a set of actual physical specifications Σ_P. Since $\hat{\varphi}_P$ and σ_P are both formulas, we construct new formulas $\hat{\varphi}_P \Rightarrow \sigma_P$ and $\sigma_P \Rightarrow \hat{\varphi}_P$, each of which may be discharged with an SMT solver. If these checks are not valid, then these specifications are candidate *cyber-physical mismatches*. These checks basically compare whether the inferred specification and actual specification are more or less restrictive than one another, in terms of the sizes of corresponding sets of states satisfying the predicates. We hypothesize that it is generally the case that the inferred physical specification should always be stronger than the actual physical specification, and only the check $\hat{\varphi}_P \Rightarrow \sigma_P$ would be needed. This would correspond to the case where the software's assumptions about the physical world are *at least as* restrictive as those made in the actual physical specification. For instance, suppose that the physical specification of the output voltage of the buck converter is $\sigma_P \triangleq t \geq t_s \Rightarrow 4.8V \leq V_{out}(t) \leq 5.2V$, and the candidate physical invariant is $\hat{\varphi}_P \triangleq t \geq t_s \Rightarrow 4.9V \leq V_{out}(t) \leq 5.1V$, then the check of the formula $\hat{\varphi}_P \Rightarrow \sigma_P$ using an SMT solver like Z3 will indicate that the system does not have a specification mismatch. Otherwise, if the candidate physical invariant is $\hat{\varphi}_P \triangleq t \geq t_s \Rightarrow 4.7V \leq V_{out}(t) \leq 5.0V$, then the check of the formula $\hat{\varphi}_P \Rightarrow \sigma_P$ will indicate that the system has a specification mismatch. On the other hand, it may also be useful to check $\hat{\varphi}_P \Leftarrow \sigma_P$, which would correspond to cases where the inferred physical specification is weaker than the actual physical specification. In this case, there may be a trace that violates the actual specification, and this may be useful in analysis like falsification to drive simulations towards a violating behavior.

[2] Z3 may be downloaded: http://z3.codeplex.com/.

5 HYNGER: GENERATING INVARIANTS FOR SLSF MODELS

Hynger—HYbrid iNvariant GEneratoR—is a software tool developed for invariant inference of CPS models represented as SLSF block diagrams[3]. Hynger is written primarily in Matlab and uses the Matlab APIs to interact with SLSF diagrams. Hynger also uses some Java code (natively inside Matlab) to interface with Daikon, which is written in Java. Daikon versions 5.0.0 to 5.1.8 were tested with Hynger[4].

Given an SLSF model $\mathcal{A}$, Hynger automatically inserts callback functions into the model to print model variables at block inputs and outputs at certain events in the SLSF simulation loop. Consequently, a trace file generated by Hynger will then be formatted in the trace input format required by Daikon. While configurable, the default behavior of Hynger is to add instrumentation (observation) points for every input and output signal for every block (recursively) in the SLSF diagram. That is, Hynger walks the tree of blocks starting from the root, and for each $v \in M$, adds instrumentation points for the input variables $\mathsf{V}_I(v)$ and the output variables $\mathsf{V}_O(v)$ of v. Of course, this may incur a drastic performance overhead, so if this is not desired, the user may select only a subset of the blocks to instrument and our performance results (see Section 6) illustrate this distinction. When an SLSF model is simulated with these instrumentation callback functions added by Hynger, it will generate a trace file in the input trace format for Daikon. Hynger also provides the capability to automatically call Daikon from Matlab (by using an appropriate Java call to Daikon), which will then return the set of candidate invariants from each program point to the user.

The Hynger flow is summarized in Figure 4. The inputs are: (a) SLSF diagrams (containing embedded software code and a set of physical variables along with their physical dynamics models [e.g., ODEs]), and (b) a set of physical variables along with their dynamics models (specified as SLSF children diagrams), and (c) a test suite for the embedded software and initial conditions for the physical simulation (such as noisy initial conditions, $\theta \in \Theta$). The output of the Hynger tool is a set of candidate invariants, which, when projected onto all the software physical variables V_{SP}, represent a candidate specification the software assumes for the physical parts of the system. Finally, candidate specifications can be checked for conformance with the actual physical requirements by comparing the two specifications: the actual physical specification and the candidate physical specification from the software perspective.

5.1 Dynamic Invariant Inference with Daikon

Next, we illustrate the dynamic invariant inference methodology used by Daikon on a pure software example. However, this pure software example (a C function) is actually specified for the controller in the buck converter case study (shown in Figure 7) in a different manner. The loop in the controller SLSF model of Figure 9 also computes a sum of an array, and Daikon can find this specification for both the SLSF controller model using Hynger, and the C-frontend for the following example. Note that, in Figure 9 the digitized output voltage from the buck-converter plant is used to determine the mode of the switch. Here, V_{tol} is denoted by the variable `Vtol`, V_{ref} is `Vref`. We highlight that the controller computes a moving average by summing an array. With Hynger and Daikon, we automatically infer that the result of this is the sum of the samples, similar to the sum return specification shown in Figure 6 found for the C function in Figure 5.

Example C Program, Formal Specification, and Candidate Invariants Inferred. Figure 5 shows an example C function to illustrate the use of dynamic analysis with Daikon to find candidate

[3] A preliminary prototype of Hynger with examples is available online: http://verivital.com/hynger/. The repository also includes Daikon input (`*.dtrace`) trace files generated from the examples, as well as the Daikon output candidate invariant (`*.inv`) files.

[4] Daikon may be downloaded: http://plse.cs.washington.edu/daikon/.

```
/*@ requires n >= 0; // at least 0 elements
  @ requires \valid(b+ (0..n-1)); // all elements exist
  @ assigns \nothing; // no side effects
  @ ensures \result == \sum(0,n-1,\lambda integer j; b[j]);
  @ ensures \result >= 0; // false , array may be negative
*/
int sum_array(int b[], unsigned int n) {
    int i;
    int s = 0;
    /*@ loop invariant
        \forall integer j; (0 <= i <= n) ==> s == \sum(0,i-1,\lambda integer j; b[j]); */
    for (i = 0; i < n; i++) {
            s += b[i];
    }
    return s;
  }
```

Fig. 5. Example C function that sums an array b of n integers. Requirements on the function inputs (i.e., preconditions on b and n for the function to be called) are specified as requires assertions in the ACSL language. Correctness specifications (i.e., postconditions following the function call) are specified as ensures assertions in the ACSL language.

```
============== Precondition
..sum_array():::ENTER
b has only one value // it's a pointer to only one location of memory
b[] elements >= 0 // all elements were non-negative for this set of traces
n == 100 // all tests were 100 element arrays for this set of traces
size(b[]) == 100 // all tests were 100 element arrays
============== Postcondition
..sum_array():::EXIT
b[] == orig(b[]) // no side effects
return == sum(b[]) // does return the sum
sum(b[]) == sum(orig(b[]))
b[] elements >= 0
```

Fig. 6. Daikon candidate invariant output (with some additional markup in C-style comments for readability) for the sum_array example from Figure 5.

invariants. The function computes and returns the sum of an array of integers. This example was recreated from an example in the original Daikon paper [17]. Additionally, a formalized correctness specification is given in the modern ANSI/ISO C Specification Language (ACSL), used by tools such as Frama-C [12]. Using Daikon and a small suite of unit tests, we were able to successfully find the invariant that returns from the function sum_array, the returned value is the sum of the elements in the array b. The suite of tests included arrays with: (*a*) all the same length and same elements, (*b*) all the same length and uniformly randomly chosen elements, (*c*) different lengths and all the same elements, and (*d*) different lengths and uniformly randomly chosen elements. Daikon successfully found the sum postcondition in all these cases with only a few test conditions. The candidate invariant outputs of Daikon appear in Figure 6, where we can see Daikon has inferred a candidate invariant that the function returns the sum of an array. We highlight that we find the sum return result of the moving average filter from Figure 9 using Hynger and Daikon.

6 EXPERIMENTAL RESULTS

Hynger was tested on Windows 10 64-bit using Matlab 2016b, and 2017a, executed on a x86-64 laptop with a 2.3 GHz dual-core Intel i5-6200U processor and 12 GB RAM. All performance metrics reported were recorded on this system using Matlab 2017a. We tested and evaluated Hynger using a number of SLSF examples, including: *(a)* the closed-loop buck converter with sensor and hysteresis controller described in Section 6.1 and detailed further in [40], *(b)* a solar array case study that uses a buck-boost converter [42], *(c)* benchmarks from S-TaLiRo [4], *(d)* benchmarks from Breach [14, 24],

Model	Solver	Tmax	Sim	SimInst	Inv	Overhead	BDAll	BDInst	BDPct
buck (Section 6.1)	ode45	0.0083	6.2985	38.4518	5.7335	7.0152	14	3	21.4286
buck (Section 6.1)	ode45	0.0083	6.4567	44.698	7.0913	8.021	14	4	28.5714
buck (Section 6.1)	ode45	0.0083	6.5301	78.3176	7.2224	13.0993	14	14	100
heat25830 [4]	ode45	50	4.6913	254.5776	14.09	57.2692	28	1	3.5714
heat25830 [4]	ode45	50	4.7328	2882.7808	15.6488	612.4233	28	10	35.7143
fuel1 [23]	ode15s	15	5.3747	976.6274	7.923	183.182	208	17	8.1731
fuel1 [23]	ode15s	15	4.2131	2824.2804	11.604	673.1137	208	63	30.2885
fuel2 [23]	ode15s	20	3.3838	36.8312	2.9881	11.7674	25	6	24
fuel2 [23]	ode15s	20	2.7353	42.4074	3.2771	16.7018	25	13	52
fuel3 [19]	ode15s	20	3.7425	292.9976	4.1131	79.3892	90	11	12.2222
fuel3 [19]	ode15s	20	3.6083	945.3992	4.3904	263.2236	90	46	51.1111

Table 1. Hynger performance results for several of the examples evaluated. Solver is the ODE solver used by SLSF. Tmax is the virtual simulation time in seconds (i.e., time from the perspective of the model). All runtime results are in seconds and are the mean of 20 runs. Sim is the simulation runtime (s). Inv is the invariant generation runtime (Daikon) (s). Overhead is the overall relative performance overhead (extra runtime) (×) using Hynger and Daikon versus only SLSF simulation (i.e., $((SimInst + Inv)/Sim)$). BDInst and BDAll are the numbers of block diagrams instrumented and the overall number of block diagrams, respectively. BDPct is the percentage (%) of block diagrams instrumented using different Hynger modes of operation (i.e., $BDInst/BDAll$).

(e) benchmarks created as a part of the ARCH 2014 CPSWeek workshop (particularly [23, 40]) and *(f)* example models provided by Mathworks. Overall, these examples vary from fairly simple with tens of blocks (such as the buck converter case study we detail), to complex (with hundreds of blocks).

Runtime Overhead from Instrumentation with Hynger and Invariant Inference with Daikon. First, we present an aggregate performance evaluation for some of these examples in Table 1, with column descriptions appearing in the caption. Overall, the performance overhead of instrumenting diagrams and performing invariant inference is around an order of magnitude increase in the best cases, and two-to-three orders of magnitude increase in the worst cases, which we note is comparable with typical Daikon instrumentation frontends like Valgrind's overhead [18, 39]. We conducted performance profiling of Hynger and identified the main source of overhead (about 75 to 90 percent) as file I/O operations. Additionally, as Hynger has several different usage scenarios and operating modes (where it may be used to instrument few blocks [subsystem and function blocks by default], many blocks [all blocks except ones such as constants, scopes, etc.], every single block, or user-selected blocks), the table illustrates these differences to give some comparison of how the methods scale on a given model. Next, we will describe two CPS case studies in details to evaluate the capability of Hynger in detecting cyber-physical specification mismatches. The first model is the closed-loop buck converter that has been used to illustrate the concepts of this paper, and the second model is derived from a collection of the automotive powertrain control benchmarks proposed by Toyota [24].

6.1 Closed-Loop Buck Converter Cyber-Physical Specification Mismatch

A basic cyber-physical specification mismatch is easy to encode in the buck converter, since the software controller inherently uses a tolerance to encode the desired output voltage ripple. This hysteresis tolerance band is typically chosen based on the system dynamics and desired output voltage ripple to ensure the output voltage meets the ripple specification. As a concrete example,

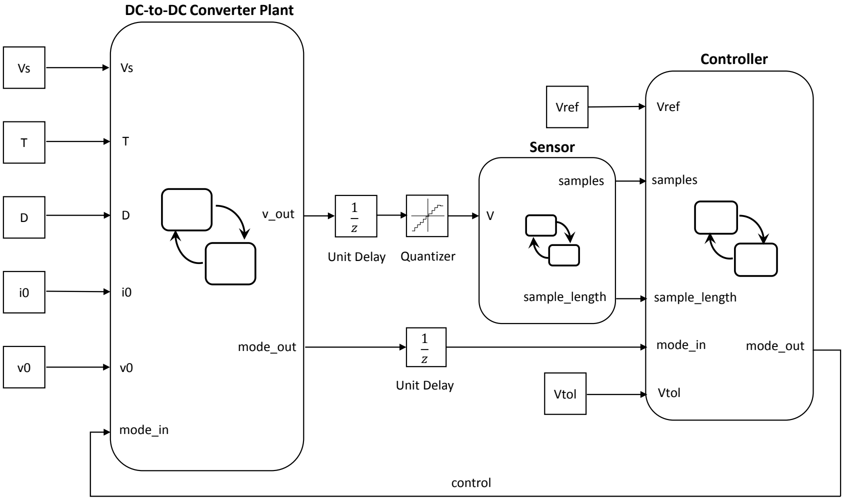


Fig. 7. General CPS case study architecture overview of the buck converter in SLSF. The system is composed of a plant (physical system) model, a controller (software/cyber), and potentially sensor and actuator models. The cyber model uses some of the physical model output states to determine a control action or input. The controller in SLSF appears in Figure 9, and the sensor model appears in Figure 8. An example of this closed-loop buck converter including only plant and controller can be formally represented as the hybrid automaton in Figure 3.

the physical specification may contain a fixed constraint that $V_{out} = V_{ref} \pm V_{rip}$, e.g., $V_{ref} = 5V$ and $V_{rip} = 0.1V$. The hysteresis band V_{tol} is then selected based on the system dynamics to ensure $4.9V \leq V_{out} \leq 5.1V$ so that it meets the requirements of the physical specifications defined by Σ_P in Section 4.1.

Sources of Cyber-Physical Specification Mismatches of the Closed-Loop Buck Converter. There are different possibilities of specification mismatch that may occur to the closed-loop buck converter. We present three scenarios that result in specification mismatches. First, if the plant parameters change (i.e., different circuit elements are used), and the software is not updated with a new hysteresis band V_{tol} to accommodate the changes in the plant dynamics, then a specification mismatch manifests. This mismatch can be detected using Hynger and the methodology described in this paper. Of course, this is a somewhat obvious mismatch, as the controller relies on variables computed as functions of the plant parameters (here, the R, L, and C values, as well as the source and desired/reference output voltage values). So if these plant components are changed, clearly the software must be updated. Second, the hysteresis controller is initially constructed using wrong information about the physical evolution of the plant. In fact, the hysteresis band V_{tol} is far different from the actual output voltage ripples V_{rip} of the plant. Third, the analog sensor of the buck converter may have ADC conversion errors that reduce the accuracy of the voltage measurement. These errors can be an offset error, a full-scale error, differential and integral non-linearity errors, etc. Moreover, a typical error that cannot be avoided in ADC sensor is the quantization error [50]. Overall, these conversion errors may cause a significant impact to result in system failures.

Experimental Results in Identifying Cyber-Physical Specification Mismatches of the Closed-Loop Buck Converter. We consider the closed-loop buck converter $\mathcal{A}$ shown in Figure 7 with $V_S = 100$, $V_{ref} = 48V$, $V_{rip} = 5\%V_{ref} = 2.4V$, and assume that δ_S, δ_{ref} are equal to zero. The physical

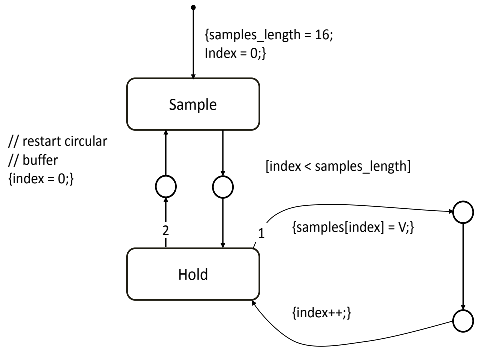


Fig. 8. Stateflow model of sensor with a sample and hold for the buck converter case study.

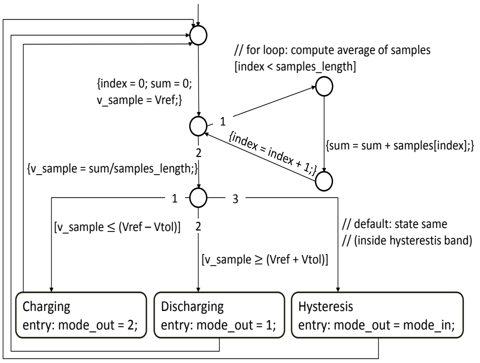

// for loop: compute average of samples
[index < samples_length]
{index = 0; sum = 0;
v_sample = Vref;}
1
2
{index = index + 1;}
{sum = sum + samples[index];}
{v_sample = sum/samples_length;}
1
3
[v_sample ≤ (Vref − Vtol)]
2
// default: state same
// (inside hysteresis band)
[v_sample ≥ (Vref + Vtol)]
Charging
entry: mode_out = 2;
Discharging
entry: mode_out = 1;
Hysteresis
entry: mode_out = mode_in;

Fig. 9. Stateflow model of the buck-converter voltage hysteresis controller.

specification of the output voltage is $\sigma_P \triangleq t \geq t_s \Rightarrow 45.6V \leq V_{out}(t) \leq 50.4V$. For the initial setup, with $R = 6\Omega$, $L = 2.65mH$, $C = 2.2mF$, and a sampling frequency $f_s = 60kHz$, the magnitude bound of the output voltage inferred from Hynger and Daikon is $\hat{\varphi}_P \triangleq t \geq t_s \Rightarrow 46.559V \leq V_{out}(t) \leq 50.203V$. Then, $\hat{\varphi}_P$ is considered as the candidate invariant of the system since the formula $\hat{\varphi}_P \Rightarrow \sigma_P$ is true. Next, we investigate different possibilities of cyber-physical specification mismatches that may occur when changing the source voltage, the desired/reference output voltage, the sampling frequency, and the plant parameters of the buck converter.

First, we increase the source voltage V_S from $100V$ to $120V$, the new magnitude bound of the output voltage inferred from Hynger and Daikon is $\hat{\varphi}_P \triangleq t \geq t_s \Rightarrow 46.804V \leq V_{out}(t) \leq 51.118V$. Then, the formula $\hat{\varphi}_P \Rightarrow \sigma_P$ is false, that indicates the system may have a cyber-physical specification mismatch.

Second, we drop the desired/reference output voltage V_{ref} to $36V$. Thus, the physical specification of the output voltage becomes $\sigma'_P \triangleq t \geq t_s \Rightarrow 34.2V \leq V_{out}(t) \leq 37.8V$. In this case, the inferred physical specification of the output voltage from Hynger and Daikon becomes $\hat{\varphi}'_P \triangleq t \geq t_s \Rightarrow 35.068V \leq V_{out}(t) \leq 39.053V$, so that the formula $\hat{\varphi}'_P \Rightarrow \sigma'_P$ is false. Therefore, changing the reference output voltage may also produce a cyber-physical specification mismatch for the buck converter.

Third, we decrease the sampling frequency f_s from $60kHz$ to $30kHz$. As a result, the new inferred physical specification of the output voltage from Hynger and Daikon is $\hat{\varphi}_P \triangleq t \geq t_s \Rightarrow 45.853V \leq V_{out}(t) \leq 51.091V$. The check of the formula $\hat{\varphi}_P \Rightarrow \sigma_P$ will return false to indicate that the system may contain a cyber-physical specification mismatch.

Next, we keep the controller unchanged and vary the values of R, L, and C to change the plant parameters. We then run the buck converter with Hynger in conjunction with Daikon, and collect candidate physical specifications associated with the output voltage. The comparison between the actual physical specification σ_P and the physical specification $\hat{\varphi}_P$ inferred from Hynger and Daikon is shown in Table 2, and also illustrated in Figure 10. Note that in Table 2, $\hat{\varphi}_P$ describes the magnitude bound of the output voltage when $t \geq t_s$. The checks of the formula $\hat{\varphi}_P \Rightarrow \sigma_P$ occasionally return $False$, that are depicted in Figure 10 when the bound of the inferred output voltage overlaps its actual bound. This indicates that changing the plant parameters without updating the controller may produce cyber-physical specification mismatches. That also proves

Parameter Values	$\hat{\varphi}_P$	$\hat{\varphi}_P \Rightarrow \sigma_P$	$\sigma_P \Rightarrow \hat{\varphi}_P$
$R = 4\Omega, L = 2.65mH, C = 2.2mF$	$45.137V \leq V_{out}(t) \leq 49.723V$	$False$	$False$
$R = 8\Omega, L = 2.65mH, C = 2.2mF$	$46.964V \leq V_{out}(t) \leq 50.405V$	$False$	$False$
$R = 6\Omega, L = 0.65mH, C = 2.2mF$	$47.141V \leq V_{out}(t) \leq 50.074V$	$True$	$False$
$R = 6\Omega, L = 6.65mH, C = 2.2mF$	$45.429V \leq V_{out}(t) \leq 50.439V$	$False$	$True$
$R = 6\Omega, L = 2.65mH, C = 1.2mF$	$45.426V \leq V_{out}(t) \leq 51.109V$	$False$	$True$
$R = 6\Omega, L = 2.65mH, C = 3.2mF$	$46.859V \leq V_{out}(t) \leq 49.774V$	$True$	$False$

Table 2. Experimental data showing the comparison between actual physical specifications and inferred physical invariants from Hynger and Daikon of the buck converter system. Here, the plant component is changed due to the changes of R, L, and C values.

the capability of Hynger and our proposed methodology in automatically detecting a candidate cyber-physical specification mismatch of CPS.

Another possibility of the specification mismatch may occur when the controller is encoded based on wrong information about the plant. For the buck converter, the hysteresis controller is built with an assumption that the output voltage ripple V_{rip} is equal to 5% of the reference voltage V_{ref}. However, the actual value of V_{rip} may be much smaller than this assumption percentage. The percentage of the output voltage ripple of the buck converter is calculated as follows [16],

$$\frac{V_{rip}}{V_{ref}} = \frac{1 - D}{8LCf_s^2}, \tag{1}$$

where $D = \frac{V_{ref}}{\eta \mathcal{V}_S}$ is a duty cycle, and η is an efficiency coefficient of the converter. Here, with $L = 2.65mH$, $C = 2.2mF$, $f_s = 60kHz$, $\eta = 0.79$, $V_{ref} = 48V$, and $\mathcal{V}_S = 100V$, the percentage of the output voltage ripple is approximately equal to 0.0002%. Thus, the hypothesized output voltage ripple used to build the controller is far larger than the actual output voltage ripple calculated by Equation 1. It definitely shows that the system may have specification mismatches since the controller is encoded depending on wrong information about the physical plant.

Furthermore, changing the length of voltage measurement array (samples_length) in the sensor of the buck converter (shown in Figure 8) may also cause a specification mismatch. For example, if we increase it from 16 to 32, the inferred physical specification using Hynger and Daikon becomes $\hat{\varphi}_P \triangleq t \geq t_s \Rightarrow 46.095V \leq V_{out}(t) \leq 50.788V$, which no longer implies the actual physical specification of the output voltage $\sigma_P \triangleq t \geq t_s \Rightarrow 45.6V \leq V_{out}(t) \leq 50.4V$.

6.2 Abstract Fuel Control System Benchmarks

In the second case study, we present the potential cyber-physical specification mismatches of the abstract fuel control (AFC) system benchmarks provided by Toyota [23, 24], and further studied in [19]. The goal of these benchmarks is to determine the fuel rate that should be injected into the manifold to maintain the air-fuel ratio within a desirable range using the feedforward and Proportional-Integral (PI) controllers. Particularly, we focus on the third model of the benchmarks including a sequence of Simulink blocks and Stateflow chart that increase levels of sophistication and fidelity of the system [19]. The model consists of four operation modes and four continuous variables. The modes include *startup*, *normal*, *power*, and *failure*; and the variables are (a) p: an intake manifold pressure, (b) p_e: an intake manifold pressure estimate, (c) λ: an air-fuel ratio, and (d) i: an integrator state, PI control signal. The evolution of the continuous variables in each mode

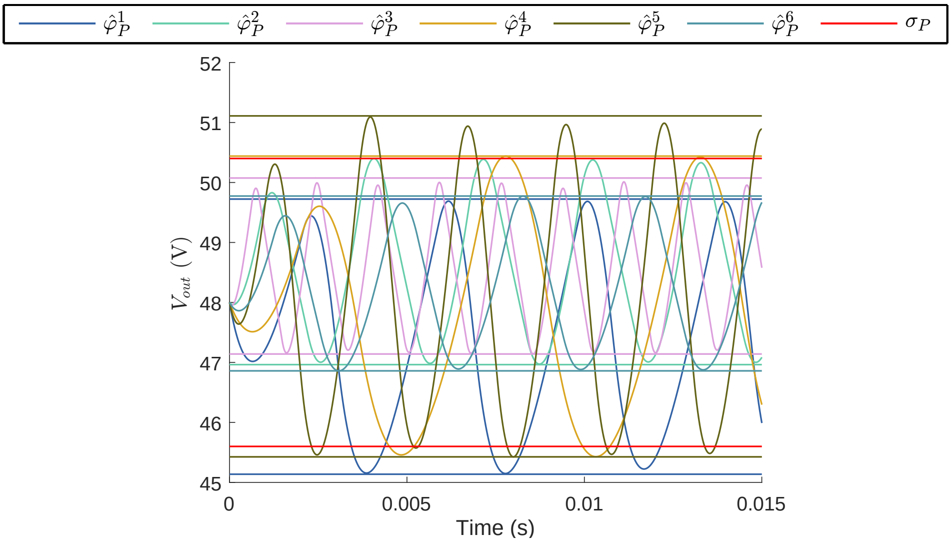


Fig. 10. A plot represents simulation traces and magnitude bounds of V_{out} of the buck converter with different values of R, L, and C. Here, σ_P denotes the actual bound of V_{out}, and $\hat{\varphi}_P^k$, $k \in [1, 6]$ denotes the inferred bound of V_{out} listed orderly in Table 2.

is governed by nonlinear polynomial differential equations as follows,

$$\dot{p} = c_1(2\theta(c_{20}p^2 + c_{21}p + c_{22}) - \dot{m}_c) \tag{2}$$

$$\dot{p_e} = c_1(2c_{23}\theta(c_{20}p^2 + c_{21}p + c_{22}) - (c_2 + c_3\omega p_e + c_4\omega p_e^2 + c_5\omega^2 p_e)) \tag{3}$$

$$\dot{\lambda} = c_{26}(c_{15} + (c_{16}c_{25}F_c + c_{17}c_{25}^2F_c^2 + c_{18}\dot{m}_c + c_{19}\dot{m}_c c_{25}F_c - \lambda) \tag{4}$$

$$\dot{i} = c_{14}(c_{24}\lambda - c_{11}), \tag{5}$$

where $F_c = \frac{1}{c_{11}}(1 + i + c_{13}(c_{24}\lambda - c_{11}))(c_2 + c_3\omega p_e + c_4\omega p_e^2 + c_5\omega^2 p_e)$, and $\dot{m}_c = c_{12}(c_2 + c_3\omega p + c_4\omega p^2 + c_5\omega^2 p)$. θ and ω are throttle angle (in degrees) and engine speed inputs (in *rpm*), respectively. The values of all constant parameters $c_j, j \in [1, 25]$, θ and ω are specified in [24]. We note that this system can be formally represented as a closed-loop CPIOA, which is the parallel composition of a plant and controller model, and both of them have three exogenous inputs including θ, ω, and sensor failure event *fail_event* [19].

AFC Plant Model. The plant can be modeled as a CPIOA with a single mode and two output physical variables p, λ whose continuous evolutions over time are described in Equation 2 and Equation 4, respectively. This model has an input cyber variable F_c, that is a fuel command.

AFC Controller Model. The controller model is a CPIOA with four operation modes including *startup*, *normal*, *power*, and *failure*. The controller has two output physical variables p_e, and i whose continuous evolutions over time are described in Equation 3 and Equation 5, respectively. Here, p and λ are considered as two input cyber variables of the controller.

Reachability analysis of a sophisticated system like the AFC system is a major contribution to both industrial and research community. However, it is a challenge to design and verify such a system using existing hybrid system verification tools. Instead, we can attempt to verify some safety requirements of the system. The AFC system has several actual physical specifications that can be found in [15]. In this section, we select two main physical specifications to evaluate the capability of Hynger and the proposed methodology. The first physical specification requires the undershoot and overshoot of the air-fuel ratio of the system should be in the settling region of ±2% of its reference value λ_{ref}. The second physical specification requires the air-fuel ratio should be maintained within ±2% of λ_{ref} in the *normal* mode when $t \geq t_s$. These properties can be formally expressed as:

$$\sigma_P^1 \triangleq mode = startup \land t \leq t_s \Rightarrow 0.98\lambda_{ref} \leq \lambda(t) \leq 1.02\lambda_{ref} \quad (6)$$

$$\sigma_P^2 \triangleq mode = normal \land t \geq t_s \Rightarrow 0.98\lambda_{ref} \leq \lambda(t) \leq 1.02\lambda_{ref}. \quad (7)$$

Initially, we set $\lambda_{ref} = 14.7$, $\theta \in [8.8°, 90°]$, $w = 1800rpm$ $t_s = 9.5s$, and the maximum simulation time $T_{max} = 20s$, the proportional and integral gains of the PI controller are $c_{13} = 0.04$ and $c_{14} = 0.14$, respectively. Next, we investigate different possibilities of cyber-physical specification mismatches for each physical specification. For the first physical specification σ_P^1, the AFC system may have specification mismatches when changing the engine speed and throttle inputs. For the second physical specification σ_P^2, the system may contain specification mismatches when changing controller and plant parameters.

Cyber-physical specification mismatches according to σ_P^1. With the initial setup mentioned earlier, the physical specification in Equation 6 becomes $\sigma_P \triangleq mode = startup \land t \leq 9.5 \Rightarrow 14.406 \leq \lambda(t) \leq 14.994$. Here, the magnitude bound of the air-fuel ratio at the *startup* mode of the system inferred from Hynger and Daikon is $\hat{\varphi}_P^1 \triangleq mode = startup \land t \leq 9.5 \Rightarrow 14.505 \leq \lambda(t) \leq 14.97$. Thus, the check of the formula $\hat{\varphi}_P^1 \Rightarrow \sigma_P^1$ is valid, that indicates $\hat{\varphi}_P^1$ is a candidate invariant of the AFC system. Next, we vary the input values and observe the consequent behaviors of the system.

First, we vary the value of the engine speed and keep other parameters unchanged. Assuming $w = 2200rpm$, the inferred physical specification of the air-fuel ratio from Hynger and Daikon becomes $\hat{\varphi}_P^1 \triangleq mode = startup \land t \leq 9.5 \Rightarrow 14.129 \leq V_{out}(t) \leq 15.033$. Hence, the formula $\hat{\varphi}_P^1 \Rightarrow \sigma_P^1$ is false indicating that the AFC system may contain a cyber-physical specification mismatch as we change the engine speed input.

Second, we change the range of the throttle input to $[40°, 70°]$. Then, the inferred physical specification of the air-fuel ratio from Hynger and Daikon becomes $\hat{\varphi}_P^1 \triangleq mode = startup \land t \leq 9.5 \Rightarrow 14.396 \leq V_{out}(t) \leq 14.849$. Hence, $\hat{\varphi}_P^1$ no longer implies σ_P^1. Therefore, there exists a cyber-physical specification mismatch when changing the throttle input as well.

Cyber-physical specification mismatches according to σ_P^2. Initially, the physical specification in Equation 7 is $\sigma_P^2 \triangleq mode = normal \land t \geq 9.5 \Rightarrow 14.406 \leq \lambda(t) \leq 14.994$. Here, the magnitude bound of the air-fuel ratio at the *normal* mode of the system inferred from Hynger and Daikon is $\hat{\varphi}_P^2 \triangleq mode = normal \land t \geq 9.5 \Rightarrow 14.645 \leq \lambda(t) \leq 14.84$. Then, we can consider $\hat{\varphi}_P^2$ as a candidate invariant of the system because the formula $\hat{\varphi}_P \Rightarrow \sigma_P$ is true.

Next, we investigate whether there is a specification mismatch for the AFC system as we change the proportional and integral gains of its PI controller. Table 3 describes the comparison between the actual physical specification σ_P^2 and the physical specification $\hat{\varphi}_P^2$ inferred from Hynger and Daikon, where $\hat{\varphi}_P^2 \downarrow \lambda$ denotes the inferred bound for λ when $t \geq t_s$ and $mode = normal$. In Table 3, the check of the formula $\hat{\varphi}_P^2 \Rightarrow \sigma_P^2$ returns false in some cases (e.g., when $c_{13} = 0.04$, $c_{14} = 0.04$) indicating that the changes in the controller gains may produce cyber-physical specification mismatches for the AFC system.

Controller Gain	$\hat{\varphi}_P^2 \downarrow \lambda$	$\hat{\varphi}_P^2 \Rightarrow \sigma_P^2$	$\sigma_P^2 \Rightarrow \hat{\varphi}_P^2$
$c_{13} = 0.01, c_{14} = 0.14$	$14.567 \leq \lambda(t) \leq 15.058$	*False*	*False*
$c_{13} = 0.02, c_{14} = 0.14$	$14.592 \leq \lambda(t) \leq 15.033$	*False*	*False*
$c_{13} = 0.06, c_{14} = 0.14$	$14.634 \leq \lambda(t) \leq 14.955$	*True*	*False*
$c_{13} = 0.8, c_{14} = 0.14$	$14.642 \leq \lambda(t) \leq 14.929$	*True*	*False*
$c_{13} = 0.04, c_{14} = 0.04$	$14.649 \leq \lambda(t) \leq 15.007$	*False*	*False*
$c_{13} = 0.04, c_{14} = 0.34$	$14.581 \leq \lambda(t) \leq 14.937$	*True*	*False*
$c_{13} = 0.04, c_{14} = 0.64$	$14.577 \leq \lambda(t) \leq 14.888$	*True*	*False*
$c_{13} = 0.04, c_{14} = 0.94$	$14.589 \leq \lambda(t) \leq 14.855$	*True*	*False*

Table 3. Experiment results illustrate the comparison between actual physical specifications and inferred physical invariants from Hynger and Daikon of the AFC system when changing the proportional gain and the integral gain of its PI controller.

7 DISCUSSION

Identifying a cyber-physical specification mismatch of CPS with dynamic analysis is a challenging problem. Although the Hynger prototype in conjunction with Daikon can detect potential cyber-physical specification mismatches of CPS, such as those in the case studies described in Section 6, however, it has some limitations. First, the Daikon tool used by Hynger may only infer extremely limited classes of nonlinear invariants by default (e.g., squares like x^2), and not general polynomials (e.g., $x^2+y^2+z^3$). So we plan to extend the invariant templates to be able to capture more interesting relations, particularly for physical variables. Second, although Daikon can infer candidate invariants in terms of logical predicates over variables, it has limitation for checking complex specifications related to real-time requirements such as STL, MTL and HyperSTL [41]. Industrial-scale CPS usually have safety and liveness requirements depending on precise real-time relations of signals, so strengthening the capability of checking temporal logic like STL, MTL and HyperSTL in Daikon would leverage the methodology presented in this paper.

Additionally, while the Hynger tool is a prototype, it can be envisioned to take an arbitrary SLSF model, instrument it, feed the resulting traces to Daikon to generate candidate invariants, then check if these candidate invariants are actual invariants or not (using, e.g., SpaceEx [20] or other hybrid system model checkers), as well as identify specification mismatches. For example, the candidate invariants inferred from Hynger and Daikon of the buck converter including only plant and controller represented in term of hybrid automata in Figure 3 would easily be checked to see whether they are actually invariants using SpaceEx. In long term, Hynger could be extended for runtime assurance tasks like detecting and thwarting security violations and attacks, similar to the ClearView tool that also uses Daikon [47]. ClearView's success for software systems illustrates that finding sets of candidate invariants and monitoring their evolution over time may be useful for runtime assurance and resiliency methods in CPS. If the candidate invariants are checked at runtime using a real-time reachability method [5], a formal and dynamic runtime assurance environment may be feasible.

8 CONCLUSION & FUTURE WORKS

The results illustrate the feasibility of using dynamic invariant inference for analysis of embedded and cyber-physical systems. The Hynger prototype enables a powerful extension of dynamic invariant inference to CPS for two main reasons. First, it enables potentially model-free and black box invariant inference, since the internals of the SLSF blocks may remain unknown. If no model is available (in the black box case), the candidate invariants represent what may be the most formal model available, otherwise (in the white box case), then candidate invariants represent a candidate abstraction of that model. If the candidate invariants are actual invariants, this is powerful, as they represent what is likely a less complex representation of the set of reachable states of the system. Second, if we view the SLSF models as hybrid automata in a formal context, it represents the first use of dynamic execution analysis for hybrid systems with sophisticated software state and discrete complexity. Two proof-of-concept CPS case studies including the DC-to-DC power converter and the powertrain fuel control system are presented to illustrate the capability of Hynger in detecting potential cyber-physical specification mismatches.

Overall, there are several directions for future research, including: (*a*) extending the classes of invariants that may be inferred, particularly to nonlinear (polynomial) [43] and disjunctive/max-plus forms [45], potentially by integrating Daikon with techniques from Dig [44], (*b*) runtime assurance and verification with real-time reachability of inferred invariants [5], (*c*) improving and refining Hynger, particularly with regard to performance (such as using Daikon in the online mode with direct pipes between Hynger and Daikon, so that file I/O is minimized), and (*d*) analyzing more industrial-scale CPS using Hynger.

ACKNOWLEDGMENTS

The material presented in this paper is based upon work supported by the National Science Foundation (NSF) under grant numbers CNS 1464311, EPCN 1509804, and SHF 1527398, the Air Force Research Laboratory (AFRL) through contract numbers FA8750-15-1-0105, and FA8650-12-3-7255 via subcontract number WBSC 7255 SOI VU 0001, and the Air Force Office of Scientific Research (AFOSR) under contract numbers FA9550-15-1-0258 and FA9550-16-1-0246. The U.S. government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of AFRL, AFOSR, or NSF.

REFERENCES

[1] 1996. *ARIANE 5 Flight 501 Failure, Report by the Inquiry Board.* Technical Report. ESA Inquiry Board, Paris, France. https://www.ima.umn.edu/~arnold/disasters/ariane5rep.html

[2] R. Alur, Thao Dang, J. Esposito, Yerang Hur, F. Ivancic, V. Kumar, P. Mishra, G.J. Pappas, and O. Sokolsky. 2003. Hierarchical modeling and analysis of embedded systems. *Proc. IEEE* 91, 1 (Jan. 2003), 11–28. DOI: http://dx.doi.org/10.1109/JPROC.2002.805817

[3] Rajeev Alur, Aditya Kanade, S Ramesh, and KC Shashidhar. 2008. Symbolic analysis for improving simulation coverage of Simulink/Stateflow models. In *Proceedings of the 8th ACM international conference on Embedded software.* ACM, 89–98.

[4] Yashwanth Annpureddy, Che Liu, Georgios Fainekos, and Sriram Sankaranarayanan. 2011. S-taliro: A tool for temporal logic falsification for hybrid systems. In *Tools and Algorithms for the Construction and Analysis of Systems.* Springer.

[5] Stanley Bak, Taylor T. Johnson, Marco Caccamo, and Lui Sha. 2014. Real-Time Reachability for Verified Simplex Design. In *IEEE Real-Time Systems Symposium (RTSS).* IEEE Computer Society, Rome, Italy.

[6] Clark Barrett, Aaron Stump, and Cesare Tinelli. 2010. The SMT-LIB Standard: Version 2.0. (2010). http://smt-lib.org/

[7] Boris Beizer. 1990. *Software testing techniques (2nd ed.).* Van Nostrand Reinhold Co., New York, NY, USA.

[8] Saddek Bensalem, Marius Bozga, Axel Legay, Thanh-Hung Nguyen, Joseph Sifakis, and Rongjie Yan. 2014. Component-based verification using incremental design and invariants. *Software & Systems Modeling* (2014), 1–25. DOI: http://dx.doi.org/10.1007/s10270-014-0410-8

[9] Francesco Bernardini, Marian Gheorghe, Francisco Jose Romero-Campero, and Neil Walkinshaw. 2007. A Hybrid Approach to Modeling Biological Systems. In *Membrane Computing*, George Eleftherakis, Petros Kefalas, Gheorghe Paun, Grzegorz Rozenberg, and Arto Salomaa (Eds.). LNCS, Vol. 4860. Springer Berlin Heidelberg, 138–159. DOI: http://dx.doi.org/10.1007/978-3-540-77312-2_9

[10] Marat Boshernitsan, Roongko Doong, and Alberto Savoia. 2006. From Daikon to Agitator: Lessons and challenges in building a commercial tool for developer testing. In *Proceedings of the 2006 international symposium on Software testing and analysis (ISSTA ’06).* ACM, New York, NY, USA, 169–180. DOI: http://dx.doi.org/10.1145/1146238.1146258

[11] C. Csallner, N. Tillmann, and Y. Smaragdakis. 2008. DySy: Dynamic symbolic execution for invariant inference. In *Software Engineering, 2008. ICSE '08. ACM/IEEE 30th International Conference on.* 281–290. DOI:http://dx.doi.org/10.1145/1368088.1368127

[12] Pascal Cuoq, Florent Kirchner, Nikolai Kosmatov, Virgile Prevosto, Julien Signoles, and Boris Yakobowski. 2012. Frama-C: A Software Analysis Perspective. In *Software Engineering and Formal Methods*, George Eleftherakis, Mike Hinchey, and Mike Holcombe (Eds.). LNCS, Vol. 7504. Springer Berlin Heidelberg, 233–247. DOI:http://dx.doi.org/10.1007/978-3-642-33826-7_16

[13] Leonardo De Moura and Nikolaj Bjørner. 2008. Z3: An Efficient SMT Solver. In *Proc. of 14th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS '08/ETAPS '08).* Springer-Verlag, 337–340.

[14] Alexandre Donzé. 2010. Breach, A Toolbox for Verification and Parameter Synthesis of Hybrid Systems. In *Computer Aided Verification*, Tayssir Touili, Byron Cook, and Paul Jackson (Eds.). Lecture Notes in Computer Science, Vol. 6174. Springer Berlin / Heidelberg, 167–170. DOI:http://dx.doi.org/10.1007/978-3-642-14295-6_17

[15] Parasara Sridhar Duggirala, Chuchu Fan, Sayan Mitra, and Mahesh Viswanathan. 2015. Meeting a Powertrain Verification Challenge. In *To appear in the Proceedings of International Conference on Computer Aided Verification (CAV 2015).*

[16] Robert W. Erickson and Dragan Maksimović. 2004. *Fundamentals of Power Electronics* (2nd edition ed.). Springer. DOI:http://dx.doi.org/10.1007/b100747

[17] M.D. Ernst, J. Cockrell, William G. Griswold, and D. Notkin. 2001. Dynamically discovering likely program invariants to support program evolution. *Software Engineering, IEEE Transactions on* 27, 2 (2001), 99–123. DOI:http://dx.doi.org/10.1109/32.908957

[18] Michael D. Ernst, Jeff H. Perkins, Philip J. Guo, Stephen McCamant, Carlos Pacheco, Matthew S. Tschantz, and Chen Xiao. 2007. The Daikon system for dynamic detection of likely invariants. *Science of Computer Programming* 69, 1–3 (Dec. 2007), 35–45.

[19] Chuchu Fan, Parasara Sridhar Duggirala, Sayan Mitra, and Mahesh Viswanathan. 2015. Progress on Powertrain Verification Challenge with C2E2. In *ARCH '15: Proc. of the 2nd Workshop on Applied Verification for Continuous and Hybrid Systems.*

[20] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. 2011. SpaceEx: Scalable Verification of Hybrid Systems. In *Computer Aided Verification (CAV) (LNCS).* Springer.

[21] D. Garlan, R. Allen, and J. Ockerbloom. 1995. Architectural Mismatch or Why it's hard to build systems out of existing parts. In *Software Engineering, 1995. ICSE 1995. 17th International Conference on.* 179–179.

[22] Shamina Hossain, Sairaj Dhople, and Taylor T. Johnson. 2013. Reachability analysis of closed-loop switching power converters. In *Power and Energy Conference at Illinois (PECI).* 130–134. DOI:http://dx.doi.org/10.1109/PECI.2013.6506047

[23] Xiaoqing Jin, Jyotirmoy V Deshmukh, James Kapinski, Koichi Ueda, and Ken Butts. 2014. Benchmarks for Model Transformations and Conformance Checking. In *1st International Workshop on Applied Verification for Continuous and Hybrid Systems (ARCH).*

[24] Xiaoqing Jin, Alexandre Donzé, Jyotirmoy V. Deshmukh, and Sanjit A. Seshia. 2013. Mining requirements from closed-loop control models. In *Proceedings of the 16th international conference on Hybrid systems: computation and control (HSCC '13).* ACM, New York, NY, USA, 43–52. DOI:http://dx.doi.org/10.1145/2461328.2461337

[25] Taylor T Johnson, Stanley Bak, and Steven Drager. 2015. Cyber-physical specification mismatch identification with dynamic analysis. In *Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems.* ACM, 208–217.

[26] Taylor T. Johnson, Zhihao Hong, and A. Kapoor. 2012. Design verification methods for switching power converters. In *Power and Energy Conference at Illinois (PECI), 2012 IEEE.* 1–6. DOI:http://dx.doi.org/10.1109/PECI.2012.6184587

[27] Kang Lee. 2000. IEEE 1451: A standard in support of smart transducer networking. In *Instrumentation and Measurement Technology Conference, 2000. IMTC 2000. Proceedings of the 17th IEEE*, Vol. 2. IEEE, 525–528.

[28] Nancy G Leveson. 2002. System safety engineering: Back to the future. *Massachusetts Institute of Technology* (2002).

[29] J. L. Lions. 1996. *Ariane 5 Flight 501 Failure.* Technical Report. Paris, France. http://www.di.unito.it/~damiani/ariane5rep.html

[30] Nancy Lynch, Roberto Segala, and Frits Vaandrager. 2003. Hybrid I/O automata. *Information and Computation* 185, 1 (2003), 105–157. DOI:http://dx.doi.org/10.1016/S0890-5401(03)00067-1

[31] Karthik Manamcheri, Sayan Mitra, Stanley Bak, and Marco Caccamo. 2011. A step towards verification and synthesis from Simulink/Stateflow models. In *Proc. of the 14th Intl. Conf. on Hybrid Systems: Computation and Control (HSCC).* ACM, 317–318. DOI:http://dx.doi.org/10.1145/1967701.1967749

[32] Kevin McCaney. 2014. Pentagon's rapid plan for maintaining air superiority. http://defensesystems.com/Articles/2014/05/01/DARPA-system-of-systems-SoSITE.aspx. (2014).

[33] Ramy Medhat, S. Ramesh, Borzoo Bonakdarpour, and Sebastian Fischmeister. 2015. A Framework for Mining Hybrid Automata from Input/Output Traces. In *Proceedings of the 12th International Conference on Embedded Software (EMSOFT*

'15). IEEE Press, Piscataway, NJ, USA, 177–186. http://dl.acm.org/citation.cfm?id=2830865.2830885

[34] Stefano Minopoli and Goran Frehse. 2016. SL2SX translator: from Simulink to SpaceEx models. In *Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control*. ACM, 93–98.

[35] Leonardo Moura and Nikolaj Bjørner. 2009. Satisfiability Modulo Theories: An Appetizer. In *Formal Methods: Foundations and Applications*, Marcel Medeiros Oliveira and Jim Woodcock (Eds.). LNCS, Vol. 5902. Springer Berlin Heidelberg, 23–36. DOI: http://dx.doi.org/10.1007/978-3-642-10452-7_3

[36] Sirajum Munir, Mohsin Y Ahmed, and John A Stankovic. 2015. EyePhy: Detecting Dependencies in Cyber-Physical System Apps due to Human-in-the-Loop. (2015).

[37] Sirajum Munir, John Stankovic, and others. 2014. DepSys: Dependency aware integration of cyber-physical systems for smart homes. In *Cyber-Physical Systems (ICCPS), 2014 ACM/IEEE International Conference on*. IEEE, 127–138.

[38] National Highway Traffic Safety Administration (NHTSA). 2011. Honda Automatic Transmission Control Module Software (Recall #11V395000). (Aug. 2011).

[39] Nicholas Nethercote and Julian Seward. 2007. Valgrind: A Framework for Heavyweight Dynamic Binary Instrumentation. In *Proceedings of the 2007 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '07)*. ACM, New York, NY, USA, 89–100. DOI: http://dx.doi.org/10.1145/1250734.1250746

[40] Luan Viet Nguyen and Taylor T. Johnson. 2014. Benchmark: DC-to-DC Switched-Mode Power Converters (Buck Converters, Boost Converters, and Buck-Boost Converters). In *Applied Verification for Continuous and Hybrid Systems Workshop (ARCH 2014)*. Berlin, Germany.

[41] Luan Viet Nguyen, James Kapinski, Xiaoqing Jin, Jyotirmoy Deshmukh, and Taylor T. Johnson. 2017. Hyperproperties of real-valued signals. In *15th ACM-IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE 2017)*. IEEE.

[42] Luan Viet Nguyen, Hoang-Dung Tran, and T.T. Johnson. 2014. Virtual Prototyping for Distributed Control of a Fault-Tolerant Modular Multilevel Inverter for Photovoltaics. *Energy Conversion, IEEE Transactions on* 29, 4 (Dec. 2014), 841–850. DOI: http://dx.doi.org/10.1109/TEC.2014.2362716

[43] ThanhVu Nguyen, Deepak Kapur, Westley Weimer, and Stephanie Forrest. 2012. Using Dynamic Analysis to Discover Polynomial and Array Invariants. In *Proceedings of the 34th International Conference on Software Engineering (ICSE '12)*. IEEE Press, Piscataway, NJ, USA, 683–693.

[44] T Nguyen, Deepak Kapur, Westley Weimer, and Stephanie Forrest. 2014. DIG: A Dynamic Invariant Generator for Polynomial and Array Invariants. *ACM Transactions on Software Engineering and Methodology, to appear* (2014).

[45] ThanhVu Nguyen, Deepak Kapur, Westley Weimer, and Stephanie Forrest. 2014. Using Dynamic Analysis to Generate Disjunctive Invariants. In *Proceedings of the 36th International Conference on Software Engineering (ICSE 2014)*. ACM, New York, NY, USA, 608–619. DOI: http://dx.doi.org/10.1145/2568225.2568275

[46] Jeremy W. Nimmer and Michael D. Ernst. 2002. Automatic generation of program specifications. In *Proceedings of the 2002 ACM SIGSOFT international symposium on Software testing and analysis (ISSTA '02)*. ACM, New York, NY, USA, 229–239. DOI: http://dx.doi.org/10.1145/566172.566213

[47] Jeff H. Perkins, Sunghun Kim, Sam Larsen, Saman Amarasinghe, Jonathan Bachrach, Michael Carbin, Carlos Pacheco, Frank Sherwood, Stelios Sidiroglou, Greg Sullivan, Weng-Fai Wong, Yoav Zibin, Michael D. Ernst, and Martin Rinard. 2009. Automatically Patching Errors in Deployed Software. In *Proceedings of the ACM SIGOPS 22Nd Symposium on Operating Systems Principles (SOSP '09)*. ACM, New York, NY, USA, 87–102. DOI: http://dx.doi.org/10.1145/1629575.1629585

[48] A Reder and A Egyed. 2013. Determining the Cause of a Design Model Inconsistency. *Software Engineering, IEEE Transactions on* 39, 11 (Nov. 2013), 1531–1548. DOI: http://dx.doi.org/10.1109/TSE.2013.30

[49] E.J. Schwartz, T. Avgerinos, and D. Brumley. 2010. All You Ever Wanted to Know about Dynamic Taint Analysis and Forward Symbolic Execution (but Might Have Been Afraid to Ask). In *Security and Privacy (SP), 2010 IEEE Symposium on*. 317–331. DOI: http://dx.doi.org/10.1109/SP.2010.26

[50] Len Staller. 2005. Understanding analog to digital converter specifications. *Embedded Systems Design* (2005).

[51] Arthur G Stephenson, Daniel R Mulville, Frank H Bauer, Greg A Dukeman, Peter Norvig, LS LaPiana, PJ Rutledge, D Folta, and R Sackheim. 1999. Mars climate orbiter mishap investigation board Phase I report, 44 pp. *NASA, Washington, DC* (1999).

[52] Stavros Tripakis, Christos Stergiou, Chris Shaver, and Edward A. Lee. 2013. A modular formal semantics for Ptolemy. *Mathematical Structures in Computer Science* 23 (8 2013), 834–881. Issue Special Issue 04. DOI: http://dx.doi.org/10.1017/S0960129512000278

[53] M.W. Whalen, A. Gacek, D. Cofer, A. Murugesan, M.P.E. Heimdahl, and S. Rayadurgam. 2013. Your What Is My How: Iteration and Hierarchy in System Design. *Software, IEEE* 30, 2 (March 2013), 54–60. DOI: http://dx.doi.org/10.1109/MS.2012.173

[54] Changyan Zhou and Ratnesh Kumar. 2012. Semantic Translation of Simulink Diagrams to Input/Output Extended Finite Automata. *Discrete Event Dynamic Systems* 22, 2 (2012), 223–247. DOI: http://dx.doi.org/10.1007/s10626-010-0096-1

Probabilistic Formal Verification of the SATS Concept of Operation

Muhammad Usama Sardar[1], Nida Afaq[1], Khaza Anuarul Hoque[2],
Taylor T. Johnson[2], and Osman Hasan[1]

[1]School of Electrical Engineering and Computer Science (SEECS)
National University of Sciences and Technology (NUST)
Islamabad, Pakistan
{usama.sardar,nida.afaq,osman.hasan}@seecs.nust.edu.pk
[2] Department of Computer Science and Engineering (CSE)
University of Texas at Arlington, USA
{khaza.hoque,taylor.johnson}@uta.edu

Abstract. The objective of NASA's Small Aircraft Transportation System (SATS) Concept of Operations (ConOps) is to facilitate High Volume Operation (HVO) of advanced small aircraft operating in non-towered non-radar airports. Given the safety-critical nature of SATS, its analysis accuracy is extremely important. However, the commonly used analysis techniques, like simulation and traditional model checking, do not ascertain a complete verification of SATS due to the wide range of possibilities involved in SATS or the inability to capture the randomized and unpredictable aspects of the SATS ConOps environment in their models. To overcome these limitations, we propose to formulate the SATS ConOps as a fully synchronous and probabilistic model, i.e., SATS-SMA, that supports simultaneously moving aircraft. The distinguishing features of our work include the preservation of safety of aircraft while improving throughput at the airport. Important insights related to take-off and landing operations during the Instrument Meteorological Conditions (IMC) are also presented.

Keywords: Formal Verification, Probabilistic Analysis, Model Checking, SATS, SATS Concept of Operations, Aircraft Safety, Aircraft Separation, Landing and Departure Operations.

1 Introduction

Small Aircraft Transportation System (SATS) [13], developed by NASA, provides access to more communities with less time delays by leveraging upon the recent advances in navigation and communication technologies. When a number of aircraft are in different parts of the airport, aircraft safety has to be ensured through timely separation and sequencing. Traditionally, non-towered non-radar airports rely on procedural separation during Instrument Meteorological Conditions (IMC), i.e., allowing only one aircraft to get access to the airport airspace at a given time, which significantly decreases the potential airport throughput

[23]. The main objective of SATS is to facilitate high volume operations (HVO) of advanced small aircraft at such airports with minimum infrastructure and low cost. Some representative SATS aircraft are Very Light Jet (VLJ) aircraft, an advanced technology Single-Engine (SE), piston-powered aircraft and an advanced technology Multi-Engine (ME), piston-powered aircraft [33].

Conventionally, SATS HVO simulations have been performed using computer programs in which aircraft modules were operated manually by pilots. These simulations develop the human-in-the-loop scenarios to check the effect of SATS procedures in the operational environment, on the pilot's responses in terms of work load and situational awareness [31,12,16,32]. In [12], off-nominal situations were also simulated, in addition to the nominal situations, to check the resulting effect on the pilot's state of mind. Proof-of-concept simulation studies were performed in the Air Traffic Control (ATC) simulation pilot lab at Federal Aviation Administration William J. Hughes Technical Center (FAATC) [30]. These simulations validated that the ATC can accept the SATS procedures, are able to control SATS traffic into and out of the Self Controlled Area (SCA), and support high volume operations. The simulations with pilots were used only for validation purposes and confirmed that SATS procedures are manageable by the airport management module (AMM). AMM's performance during high arrival rates of aircraft into the SCA has also been studied and found to have less delays as compared to one-in-one-out method [27]. Recently, an algorithm has been developed to optimize SATS landing sequence for multiple aircraft in [4], to make it conflict-free and with less delays, using Microsoft VC++ 6.0 simulation environment. However, these piloted simulation methods lack exhaustiveness [14] in terms of coverage of all the possible states as a rigorous piloted simulation of all possible scenarios requires a large number of tests, which in turn demands a significant amount of computational power and time. This leads to another major challenge of simulation-based verification of the SATS Concept of Operations (ConOps), i.e., selection of test vectors. A random selection of test vectors cannot offer a guarantee of correctness of the SATS ConOps since it might miss the meaningful portion of the design space. Moreover, it may not be possible to consider or even foresee all corner cases. Consequently, simulation-based verification of the SATS ConOps is incomplete with respect to error detection, i.e., all errors in a system cannot be guaranteed to be detected, which is a severe limitation considering the safety-critical nature of passenger aircraft.

In order to have a complete analysis, automatic parameterized verification of hybrid automata [20,19] was recently employed to verify properties of the SATS ConOps using model checking principles, while considering position of the aircraft as a continuous variable modeled either as a timer [19] or as a rectangular differential inclusion [20]. While this methodology allows for verification regardless of the number of aircraft, a limitation of this work is that the methodology requires the user to specify inductive invariants sufficient to establish safety. While the process of finding inductive invariants sufficient to establish safety of the SATS ConOps has been successfully automated through an extension of invisible invariants [3], this is an incomplete (heuristic) method that, in general, may fail to find such inductive invariants [21]. The analysis and formal verification of the timing constraints of SATS was done in [10] using Linear Real-Time Logic (LRTL). The higher-order-logic theorem prover PVS [26] has also been used for the safety verification of the SATS ConOps [13,9,23,29]. In particular, it has been formally verified that SATS rules and procedures can provide minimum required spacing between two and more aircraft. A hybrid modeling technique was also developed in PVS using the PVS tool Besc [25].

In the above-mentioned methods of validation and verification of SATS, only the procedures and transition rules are considered. With these considerations, any model with appropriate conditions can validate that the procedures are enough for the assurance of safe separation between the aircraft. The missed approach transition is dependent on many random factors, for instance, low visibility. In conventional airports, it is mainly caused by the bad weather, increased air-borne traffic density, and ground traffic and its delays [15]. It is also required upon the execution of a rejected landing because of objects, such as men, equipment or animals, on the runway [1]. Due to such uncertainties involved, it is necessary to incorporate the probabilistic considerations of the system into the validation methods and safety verifications of SATS. Hence, we propose to use probabilistic model checking [5,11] for the verification of the SATS ConOps. This paper presents a fully synchronous Discrete-Time Markov Chain (DTMC) model of the SATS ConOps and the verification of the safety properties of SATS, including the landing and take-off procedures, using the probabilistic model checker PRISM [22]. PRISM has been extensively used to formally model and analyze a wide variety of systems, including communication and multimedia protocols, randomised distributed algorithms, security protocols, biological systems and many others, that exhibit random or probabilistic behaviour [2].

The rest of the paper is organized as follows: Section 2 describes the SATS operational concept to facilitate the understanding of the rest of the paper. Section 3 explains the main challenges that we faced in modeling the considered, fully synchronous, system in PRISM and the assumptions used in our DTMC model. In this section, our modeling methodology is also explained through discussion about each module, transition rules and procedures. Section 4 presents the probabilistic verification results of the SATS ConOps and the novel observations made. Finally, Section 5 concludes this paper by drawing conclusions and mentioning some directions of future work.

2 SATS ConOps

The ConOps for SATS is primarily a set of rules and procedures based on an area surrounding the airport, called the SCA, a centralized automated system, called the AMM, data communication between AMM and aircraft and state data broadcast from the aircraft [8,7]. The SCA is typically taken as a region with 12-15 nautical miles radius and 3000 feet above the ground [8,9]. It is arranged in a T structure, consisting of base, intermediate and final zones. It is divided into a number of segments and fixes which are the latitude/longitude points in space.

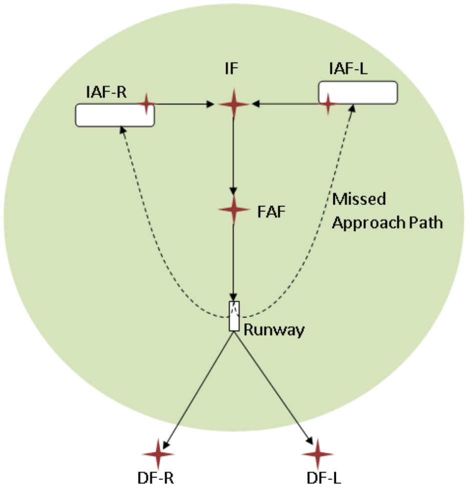


Fig. 1: Top view of the SCA [13]

The fixes are initial arrival fixes (IAFs), intermediate fix (IF), final approach fix (FAF) and departure fixes (DFs), as shown in Fig. 1. The IAFs serve two purposes, i.e., holding fix, when an aircraft enters the SCA, and missed approach holding fix (MAHF), which is required when an aircraft misses landing, and flies back to the IAF via missed approach path.

There are two types of entries into the SCA: vertical entry and lateral entry [9,25], as depicted in Fig. 2. Vertical entry is always made from the 3000 feet holding fix at the left (above IAF-L) or right (above IAF-R). Thereafter, the aircraft descends to the respective 2000 feet holding fix when it becomes available. Next, under certain conditions, the aircraft moves to the base segment (IAF to IF). On the other hand, in a lateral entry, the aircraft flies from the point of entry to the base segment directly or through the 2000 feet holding fix. Once the aircraft is in the base segment or 2000 feet holding fix, there is no dependency on its type of entry. After base segment, the aircraft goes through the IF, FAF, and finally reaches the runway. This procedure is primarily composed of a series of transitions through different segments of the SCA that are conducted by the aircraft if sufficient separation from the other aircraft is available and all conditions for the given transitions hold. If an aircraft misses its landing, due to any reason, it has to follow the missed approach path to move to the IAF corresponding to its MAHF assignment, as shown in Fig. 1.

The AMM has the responsibility to grant permissions to the aircraft for entering the SCA [7,31]. While granting the permission, the AMM assigns a landing sequence and a MAHF to the aircraft. These landing sequence numbers encode the leader information and also identify whether an aircraft is the first aircraft in a specific zone of SCA. The aircraft entering later thus follows the leader during the transitions. The MAHF assignment is in terms of ‘side’, which

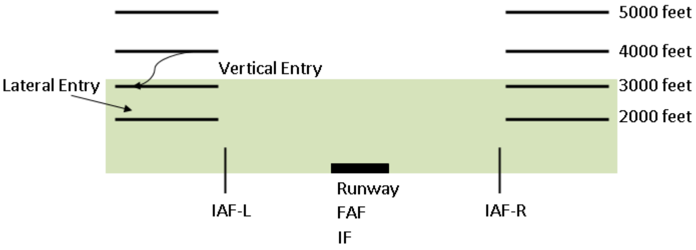


Fig. 2: Side view of the SCA [13]

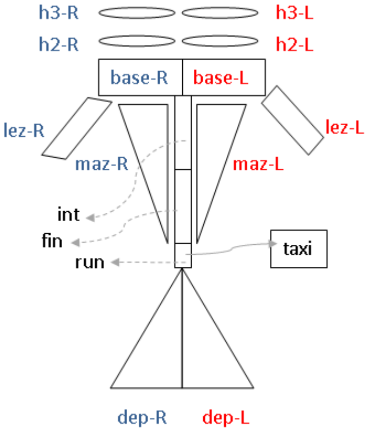


Fig. 3: Zones of the SCA [13]

can assume values of right or left. If the entering aircraft is the first one in sequence, then its MAHF will be in the same side from which it is entering. Whereas, the next aircraft, with sequence other than 1, will have the MAHF that is opposite to that of its leader.

Departure fixes are outside the SCA and under the ATC control. An aircraft ready to depart requests ATC for clearance. After clearance, the departure operation starts at the runway and it moves to the departure fix corresponding to its MAHF assignment. A safe distance of 10 or 5 nautical miles has to be maintained from the aircraft flying to the same or opposite departure fixes, respectively [13].

The SCA can be divided into different zones, illustrated in Fig. 3 and presented in Table 1. These zones represent the state of the aircraft. The complete information about the aircraft will thus include the sequence and MAHF assigned by AMM and the current location/zone of aircraft. The safety verification is based on the number of aircraft in a zone and their separation from other aircraft in other zones [23].

Table 1: Zones of SCA [13]

Zone	Symbol	Description
1	h3-R	Holding at 3000 feet at right side
2	h3-L	Holding at 3000 feet at left side
3	h2-R	Holding at 2000 feet at right side
4	h2-L	Holding at 2000 feet at left side
5	lez-R	Lateral entry zone at right side
6	lez-L	Lateral entry zone at left side
7	base-R	Right segment of base (IAF-R to IF)
8	base-L	Left segment of base (IAF-L to IF)
9	int	Intermediate segment (IF to FAF)
10	fin	Final segment (FAF to runway)
11	run	Runway
12	maz-R	Missed approach zone at right of base
13	maz-L	Missed approach zone at left of base
14	taxi	Taxi
15	dep-R	Right departure path towards right departure fix
16	dep-L	Departure path towards left departure fix

3 Formal Modeling of SATS as a DTMC in PRISM

In this section, we first describe our refinements to the SATS ConOps. Then the main challenges encountered in modeling the system in PRISM are presented. This is followed by the description of how these challenges were tackled in our model.

3.1 Refinements to original SATS

The proposed model of the SATS ConOps in the PRISM language overcomes some of the limitations of the non-deterministic, asynchronous transition system presented by Dowek et. al [13]. Before presenting the details of our model, we find it appropriate to point out the discrepancies in the existing algorithm and our proposed solution.

1. In a non-deterministic model, if two or more rules are enabled simultaneously, any one of them is allowed to be executed. In other words, only one non-deterministic action happens at a time. This means that in such a model, at each time step, *only one* aircraft will move to the next zone while all other aircraft hold in the same zone, even if the conditions are satisfied for all aircraft to move to their respective next zones. Thus, one aircraft could change zones several times while another remains idle [13]. Hence, such a model is unrealistic [23], as it fails to depict the real scenario.
2. The lowest available altitude determination (Rule 12) [13] is a simultaneous transition, potentially involving 2 aircraft, when the holding pattern at 3000 feet is occupied but 2000 feet is available. In this case, the transition

determines 3000 feet as the lowest available altitude and forces the aircraft holding at 3000 feet to descend to the holding pattern at 2000 feet. This is a weakness of the model because simultaneous transition is not possible in a fully non-deterministic model.

Our proposed solution for both the above limitations is to build a fully synchronous model that allows simultaneously moving aircraft. Hence, at each time step, all aircraft satisfying conditions to move to their respective next zones are allowed to proceed concurrently. Moreover, this model also facilitates the simultaneous transition in the lowest available altitude determination.

3.2 Modelling Challenges of SATS in PRISM

Parallel Composition of Modules
Parallel composition of modules in PRISM may seem to be the best option for developing the interleaved model of concurrency of aircraft in the SCA, where each module represents an aircraft. However, there are critical limitations in such a model, as discussed in Section 3.1. When multiple commands (belonging to any of the modules) are enabled at the same time, the choice between which command is executed by PRISM is *non-deterministic* in case of Markov decision process (MDP) and *probabilistic* in case of DTMC [2]. Specifically in the case of a DTMC, PRISM selects the command for execution uniformly at random. For instance, if there are 4 aircraft in the SCA and guards are satisfied for one command in each module, then there is a probability of 0.25 for each aircraft to move forward to the next zone. But only one of them is selected to move at a time.

Synchronization
PRISM supports synchronized transitions using synchronization labels. In this case, commands can be labelled with actions, which can be used to force two or more modules to make transitions simultaneously. By default, all modules are combined using the standard CSP parallel composition, i.e., modules synchronize over all their common actions [2]. However, in SATS application, the aircraft can be in any of the 16 zones and thus only a *specific scenario* can be modelled using synchronization labels. For instance, if there are two aircraft and the command for the first aircraft to be in the third zone is synchronized with the command for the second aircraft to be in the first zone, then they will make the transition simultaneously, if available, but it models a special case out of the many possibilities. They will no longer be synchronized in some future time step when the first aircraft is, for instance, in the seventh zone while the second aircraft is in the first zone.

Global variables with Synchronization
Global variables seem useful in modelling the state of the aircraft in the SCA as, unlike local variables, they are modifiable from any module. However, an

important restriction on the use of global variables in PRISM is the fact that global variables cannot be updated on a synchronized command [2]. PRISM detects this and reports an error if an attempt is made to do so.

Probabilistic Updates

In order to correctly model the semantics of the communication between aircraft and AMM, both aircraft and AMM should have separate modules in PRISM. Unfortunately, there is no direct way of changing a variable in a different module for *only one* probabilistic update of a command in the *same* time step. However, such probabilistic updates are frequently required. For instance, when an aircraft is in the final zone and it can move to the runway or missed approach path with certain probabilities. In case a pilot chooses the missed approach path, a new sequence number is to be assigned to the aircraft by the AMM while in case of transition to runway, there is no change in the sequence number. A possible solution could be to change the model such that the relevant variable is part of the same module as the probabilistic update but it will not represent the actual scenario of the communication between aircraft and the AMM.

Therefore, the challenge is to achieve a synchronization such that all aircraft move together whenever the guard conditions are satisfied, while incorporating probabilistic updates from the AMM in the model.

3.3 Modeling SATS in PRISM

In our formal model [28], we formulate the SATS ConOps as a DTMC in the PRISM model checker using an abstract timing model. Both sides of the approach are symmetric [13,29] and there can be at most two aircraft on each side of the SCA [13,23]. Therefore, we have assumed two aircraft in the right side of the SCA in this work for the purpose of simplicity. Our model ensures that after a landing aircraft has landed safely, it unloads passengers of the current flight in the taxi state. Then, it loads passengers of the next flight and is ready for departure. After departure, it reaches its destination and the next time it becomes a landing aircraft for the SCA. Hence, the process of landing and departure continues.

Model of Concurrency

In order to cope with the challenges, described in Section 3.2, we modeled the SATS ConOps as fully synchronously parallel automata, as in [17], where each transition is labeled with the same synchronization label, and therefore at each time step, at least one transition of each module is active. Hence, in such a fully synchronous model, both aircraft move concurrently to the next respective zones whenever the conditions are satisfied. In order to use the same synchronization label `t` with all commands in all modules, we ensure that *at least* one condition is true for each module for each reachable state in our model.

Model of SATS Transition Rules and Procedures

The modules `aircraft1` and `aircraft2` in our formal model [28], corresponding to each aircraft, implement the rules of ConOps, i.e., under what conditions the aircraft moves from one zone to the next. The modules are symmetric except that priority is assigned to `aircraft1` in case of simultaneous entry. Due to our proposed fully synchronous model, aircraft can enter inside the SCA individually or simultaneously with another aircraft. The state variables `zone1` and `zone2` represent the current zone of `aircraft1` and `aircraft2`, respectively. They are modelled as integer variables with values in the range 0 - 16, and the encoding is listed in Table 1. One additional zone is to be included into the model, which is the 'fly zone', for an aircraft outside the SCA. We encode it with a value of zero. In our model, we used formulas for compact representation of the conditions and to avoid repetition. For instance, `z1_total` represents the total number of aircraft in zone 1 and `z7_total_R` represents number of aircraft in zone 7 with an MAHF assignment of right, as shown in the following lines of the code in PRISM language:

$$formula\ z1\_total = (zone1 = 1?1 : 0) + (zone2 = 1?1 : 0);$$

$$\begin{aligned} formula\ z7\_total\_R = {} & (zone1 = 7\ \&\ mahf1 = true?1 : 0) \\ & + (zone2 = 7\ \&\ mahf2 = true?1 : 0); \end{aligned}$$

Model of the AMM

The AMM is the sequencer of the SCA. It typically resides at airport ground and communicates with the aircraft via a data link [8]. We model AMM as a separate module `AMM` in PRISM to represent this communication with the aircraft. It has two state variables, i.e., `seq` and `mahf` for each aircraft. For a landing aircraft, `seq` represents the relative landing sequence number, such that the aircraft with landing sequence *n* is the leader of the aircraft with landing sequence *n+1*, i.e., an aircraft with sequence number 1 is leader of the aircraft with sequence number 2. It is modelled as an integer variable with values in the range 0 - 10. When an aircraft enters the SCA, `seq` is assigned a new value calculated by the formula `nextseq`. This value is calculated based on the number of the aircraft already in the landing zones of the SCA. In case of simultaneous entry by both aircraft, different sequence numbers are assigned to both the aircraft, with priority to `aircraft1`. A new sequence number is also assigned when an aircraft initiates a missed approach path and the sequence numbers of all other aircraft in the landing zones of the SCA are decremented by one. Moreover, when an aircraft enters runway, the sequence numbers of all other aircraft in the SCA are again decremented by one. When an aircraft moves to the taxi state, its sequence number becomes 0. For a departing aircraft, `seq` represents the distance of the aircraft from runway in nautical miles. It is incremented by one in each time step when it is in one of the departure zones, until it becomes 10, where it is assumed to have left the SCA. The MAHF of an aircraft, represented by `mahf`, is a boolean variable with `true` representing right MAHF, and `false` representing left MAHF. It is assigned whenever an aircraft enters the SCA. Moreover, it is re-assigned when an aircraft executes a missed path approach. We consider MAHF of only right side for simplicity of the model in this paper.

Timing Model
We use an abstract timing model in our formalization of the SATS ConOps. We assume that each aircraft stays in a zone for at least one time step. So, an aircraft must transition to the next zone after one time unit if the conditions for transition are satisfied. When the guard conditions are not fulfilled, it stays in the zone until the conditions become true.

Randomness in Model
Since there is no direct way of changing a variable in a different module for only one probabilistic update of a command in the *same* time step, we introduce an additional chooser module for each probabilistic decision. For instance, consider an aircraft in the final zone. Now it can either choose the missed approach path with a probability `p_map` or it can continue landing and transit to the runway with probability `1-p_map`. In case of the missed approach path, a new sequence number and MAHF is to be assigned to the aircraft. However, there is no change in its sequence number and MAHF if it proceeds to runway. We propose to use the chooser module, `choose_p_map` which contains a single state variable `p_map_state` of type integer and with two possible values: 0 and 1. When the probability `p_map` is selected, `p_map_state` is set to 1, otherwise it is 0. This is achieved by using the following command in PRISM:

$$[t]\ Guard \rightarrow p\_map : (p\_map\_state' = 1) + (1 - p\_map) : (p\_map\_state' = 0);$$

It is important to note that instead of setting `true` as a guard, we use the conditions of transition to final zone, i.e., one step back condition as the guard [28]. This way, the command does not execute on each time step. `p_map_state` is updated when the aircraft enters the final zone and is ready to be used when checking conditions for the next transition to runway or missed approach zone in the next time step.

The value of `p_map_state` is now used in such a way that the guard condition of `p_map_state=1` checks whether `p_map` is selected. For instance, in the `AMM` module, the following command ensures that `seq1` and `mahf1` are updated as soon as it makes the transition to zone 12:

$$[t]\ Guard\ \&\ p\_map\_state = 1 \rightarrow (seq1' = nextseq)\ \&\ (mahf1' = nextmahf1);$$

4 Verification Results

4.1 Safety Properties

Based on our model, explained in Section 3, safe separation is not maintained when two aircraft reside simultaneously in the specific zones. These zones include the approach, final approach, missed approach, runway and departure zones. Hence, we label this state `danger` as follows:

$$\begin{aligned} label\ \text{“}danger\text{”} = ((zone1 = 7 \& zone2 = 7) \mid (zone1 = 9 \& zone2 = 9) \\ \mid (zone1 = 10 \& zone2 = 10) \mid (zone1 = 11 \& zone2 = 11) \\ \mid (zone1 = 12 \& zone2 = 12) \mid (zone1 = 15 \& zone2 = 15)); \end{aligned}$$

Safety in all Paths: $P =?\ [F\ \text{“}danger\text{”}];$

> We analyze safety in our model using the above property, which computes the value of the probability that `danger` is satisfied in the future by the paths from the initial state. PRISM shows a result of 0, which confirms that no path leads to a collision from the initial state.

Safety in all Reachable States: $filter\ (forall, P <= 0\ [F\ \text{“}danger\text{”}]);$

> In order to confirm that the probability of occurrence of `danger` remains 0 for all *reachable* states, we formalize the property using filters as above. The property verifies to be true in PRISM and thus guarantees the safety in our model.

4.2 Analysis of Landing and Departure Operations

Expected Time for Landing: $R =?\ [F\ \text{“}landings1\text{”}];$

We utilize the *reachability* reward [2] in PRISM to find the *expected* time taken for the landing of an aircraft in our model. In this case, a reward of unity is awarded to each state of the model and the rewards are accumulated along a path until a certain point is reached. We define this point as the state in which the aircraft is in the taxi state, for instance, for `aircraft1`:

$$label\ \text{“}landings1\text{”} = (zone1 = 14);$$

Since very limited information is available on the probability of executing a missed approach path `p_map` for SATS, we leverage upon the PRISM's parametric model checking functionality to perform the sensitivity analysis on the values of `p_map` from 0.001 to 0.9. The results are shown in Fig. 4, which depict the exponential increase in the expected time taken for landing with `p_map`. Since `aircraft1` is assigned priority in case of simultaneous entry, the values for this aircraft are slightly smaller as compared to those of `aircraft2`. The overall expected time for any aircraft to land is also shown.

Expected Number of Departures in a Fixed Time: $R =?\ [C\ <= T];$

We leverage upon the *cumulative* reward properties [2] to find the *expected* number of departures of the aircraft in a fixed time in our model. In this case,

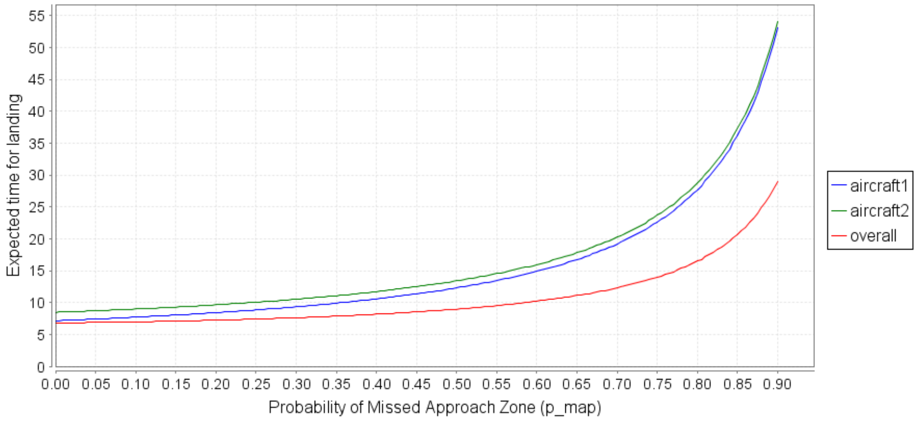


Fig. 4: Expected time for landing vs. Probability of the Missed Approach Zone

a reward of unity is awarded to each transition of departure and the rewards are accumulated until `T` time steps have elapsed. Fig. 5 shows the results of an experiment with `T` set to 10,00,000 which is large enough for the purpose of comparative analysis. Since `aircraft1` is assigned priority in case of simultaneous departure, the expected number of departures for this aircraft are slightly larger as compared to those of `aircraft2`.

Comparison of SATS and SATS-SMA: Reproduction of the corresponding non-deterministic model [13] in PRISM shows that the expected number of landing or departure operations are much greater in our proposed SATS-SMA than the corresponding non-deterministic model. For instance, with no aircraft executing a missed approach path, i.e., `p_map` of 0, the *expected* operations in the original non-deterministic asynchronous model and our refined SATS-SMA are 51280 and 81081, respectively, i.e., around 1.6 times greater throughput. The reason is that original SATS allows only one aircraft to move at a time while we allow all aircraft satisfying the conditions to move simultaneously to the respective next zones.

The key advantages of this work include the increase in the throughput, while maintaining aircraft safety, through simultaneous operations. The work also provides important quantitative landing and departure insights of the SATS ConOps. Our PRISM code and properties file is available for download [28], and thus can be benefited by researchers and verification engineers for further developments and analysis of the SATS ConOps.

5 Conclusion

Given the random and unpredictable nature of entry of aircraft into the SCA and transitions between the zones, we propose to use a probabilistic model checker,

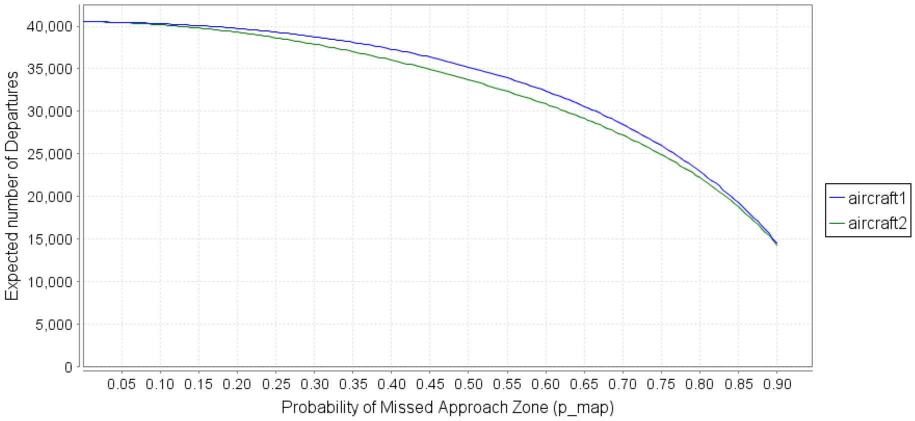


Fig. 5: Expected departures vs. Probability of the Missed Approach Transition

PRISM, to analyze the SATS ConOps in this paper. A fully synchronous DTMC model of SATS is proposed and is verified to increase the expected throughput of the airport as compared to the traditional non-deterministic, asynchronous model. Moreover, the successful modeling and verification of the transition procedures for two aircraft moving concurrently, has verified the safety of aircraft in terms of safe separation in all zones including take-off and landing. The landing and departure operations of SATS are analyzed with respect to the probability associated with the missed approach transition.

An important direction of future work is to improve the timing model by incorporating zone distances and abstract aircraft kinematics [25]. A more detailed analysis can be carried out by removing the simplifying assumptions of 2 aircraft and right side MAHF. Similarly, detailed comparison of non-SATS (one-in/one-out), SATS and SATS-SMA is an interesting direction for future research. Furthermore, we also plan to conduct the probabilistic analysis of the SATS ConOps under off-nominal conditions [24,6,12], such as equipment malfunction and emergency situations, using the parametric model checking functionality of PRISM, like it was utilized for the analysis of probability of missed approach in this paper. Moreover, Continuous-Time Markov Chains (CTMCs) of the SATS ConOps can also be developed to verify some time-related properties, where Erlang distribution can be used to model discrete time delays [18].

Acknowledgments. We would like to express our profound gratitude and heartfelt thanks to Dr. Cesar A. Munoz from NASA Langley Research Center for the valuable insights related to SATS and their model. We are also enormously pleased to precise our intense gratefulness and deepest gratitude to Dr. Matthias Gudemann for his helpful tips on modeling the system in PRISM. K. A. Hoque and T. T. Johnson are supported in part by the National Science Foundation (NSF) via grant number CNS 1464311, the Air Force Research

Laboratory (AFRL) via contract number FA8750-15-1-0105, and the Air Force Office of Scientific Research (AFOSR) via contract number FA9550-15-1-0258. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of AFRL, AFOSR, or NSF.

References

1. Instrument Procedures Handbook. U.S. Department of Transportation, Federal Aviation Administration (2015)
2. PRISM - Probabilistic Symbolic Model Checker. `http://www.prismmodelchecker.org` (2016)
3. Arons, T., Pnueli, A., Ruah, S., Xu, Y., Zuck, L.: Parameterized verification with automatically computed inductive assertions? In: Computer Aided Verification, vol. 2102, pp. 221–234. Springer (2001)
4. Bai, C., Zhang, X.: Aircraft landing scheduling in the small aircraft transportation system. In: International Conference on Computational and Information Sciences. pp. 1019–1022. IEEE (2011)
5. Baier, C., Katoen, J.P., et al.: Principles of model checking, vol. 26202649. MIT Press (2008)
6. Baxley, B., Williams, D., Consiglio, M., Adams, C., Abbott, T.: The small aircraft transportation system (SATS), higher volume operations (HVO) off-nominal operations. In: Aviation, Technology, Integration, and Operations Conference. American Institute of Aeronautics and Astronautics (2005)
7. Baxley, B., Williams, D., Consiglio, M., Adams, C., Abbott, T.: Small aircraft transportation system, higher volume operations concept and research summary. Journal of Aircraft 45(6), 1825–1834 (2008)
8. Carreño, V.: Concept for multiple operations at non-tower non-radar airports during instrument meteorological conditions. In: Digital Avionics Systems Conference. vol. 1, pp. 5.B.1–5.1–9. IEEE (2003)
9. Carreño, V., Muñoz, C.: Safety verification of the small aircraft transportation system concept of operations. In: Aviation, Technology, Integration, and Operations Conference. American Institute of Aeronautics and Astronautics (2005)
10. Cheng, A., Niktab, H., Walston, M.: Timing analysis of small aircraft transportation system (SATS). In: Conference on Embedded and Real-Time Computing Systems and Applications. pp. 58–67. IEEE (2012)
11. Clarke, Jr., E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press (1999)
12. Consiglio, M., Conway, S., Adams, C., Syed, H.: SATS HVO procedures for priority landings and mixed VFR/IFR operations. In: Digital Avionics Systems Conference. vol. 2, pp. 13.B.2–1–13.B.2–8. IEEE (2005)
13. Dowek, G., Munoz, C., Carreño, V.A.: Abstract model of the SATS concept of operations: Initial results and recommendations. Tech. Rep. NASA/TM-2004-213006, NASA Langley Research Center (2004)
14. Fedeli, A., Fummi, F., Pravadelli, G.: Properties incompleteness evaluation by functional verification. IEEE Transactions on Computers 56(4), 528–544 (2007)
15. Gariel, M., Spieser, K., Frazzoli, E.: On the statistics and predictability of go-arounds. In: Conference on Intelligent Data Understanding (2011)
16. Greco, A., Magyarits, S., Doucett, S.: Air traffic control studies of small aircraft transportation system operations. In: Digital Avionics Systems Conference. vol. 2, pp. 13.A.4–1–13.A.4–12. IEEE (2005)

17. Güdemann, M., Ortmeier, F.: A framework for qualitative and quantitative formal model-based safety analysis. In: Symposium on High-Assurance Systems Engineering. pp. 132–141. IEEE (2010)
18. Hoque, K.A., Mohamed, O.A., Savaria, Y.: Towards an accurate reliability, availability and maintainability analysis approach for satellite systems based on probabilistic model checking. In: Design, Automation Test in Europe Conference Exhibition. pp. 1635–1640. IEEE (2015)
19. Johnson, T.T., Mitra, S.: Parameterized verification of distributed cyber-physical systems: An aircraft landing protocol case study. In: International Conference on Cyber-Physical Systems. pp. 161–170. IEEE (2012)
20. Johnson, T.T., Mitra, S.: A small model theorem for rectangular hybrid automata networks. In: Joint International Conference on Formal Methods for Open Object-Based Distributed Systems and Formal Techniques for Networked and Distributed Systems. pp. 18–34. Springer (2012)
21. Johnson, T.T., Mitra, S.: Invariant synthesis for verification of parameterized cyber-physical systems with applications to aerospace systems. In: Infotech at Aerospace Conference. American Institute of Aeronautics and Astronautics (2013)
22. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: Computer Aided Verification. vol. 6806, pp. 585–591. Springer (2011)
23. Muñoz, C., Dowek, G., Carreño, V.: Modeling and verification of an air traffic concept of operations. Software Engineering Notes 29(4), 175–182 (2004)
24. Muñoz, C., Carreño, V., Dowek, G.: Formal analysis of the operational concept for the small aircraft transportation system. In: Rigorous Development of Complex Fault-Tolerant Systems, vol. 4157, pp. 306–325. Springer (2006)
25. Muñoz, C., Dowek, G.: Hybrid verification of an air traffic operational concept. In: IEEE ISoLA Workshop on Leveraging Applications of Formal Methods, Verification, and Validation (2005)
26. Owre, S., Rushby, J.M., Shankar, N.: PVS: A prototype verification system. In: Conference on Automated Deduction, pp. 748–752. Springer (1992)
27. Peters, M.: Capacity analysis of the NASA Langley airport management module. In: Digital Avionics Systems Conference. vol. 1, pp. 4.D.6 – 41–12. IEEE (2005)
28. Sardar, M.U., Hoque, K.A.: Probabilistic formal verification of the SATS concept of operation. `http://save.seecs.nust.edu.pk/projects/SATS` (2016)
29. Umeno, S., Lynch, N.: Proving safety properties of an aircraft landing protocol using I/O automata and the PVS theorem prover: A case study. In: International Symposium on Formal Methods, pp. 64–80. Springer (2006), `http://dx.doi.org/10.1007/11813040_5`
30. Viken, S.A., Brooks, F.M.: Demonstration of four operating capabilities to enable a small aircraft transportation system. In: Digital Avionics Systems Conference. vol. 2, pp. 13.A.1–1–13.A.1–16. IEEE (2005)
31. Williams, D.M.: Point-to-point! validation of the small aircraft transportation system higher volume operations concept. In: International Congress of Aeronautical Sciences (2006)
32. Williams, D., Consiglio, M., Murdoch, J., Adams, C.: Flight technical error analysis of the SATS higher volume operations simulation and flight experiments. In: Digital Avionics Systems Conference. vol. 2, pp. 13.B.1–1–13.B.1–12. IEEE (2005)
33. Xu, Y., Baik, H., Trani, A.: A preliminary assessment of airport noise and emission impacts induced by small aircraft transportation system operations. In: Aviation Technology, Integration and Operations Conference. American Institute of Aeronautics and Astronautics (2006)

Non-linear Continuous Systems for Safety Verification (Benchmark Proposal)

Andrew Sogokon[1], Khalil Ghorbal[2], and Taylor T. Johnson[3]

[1] University of Texas at Arlington, Arlington, TX, USA
andrew.sogokon@uta.edu
[2] INRIA, Rennes, Brittany, France
khalil.ghorbal@inria.fr
[3] University of Texas at Arlington, Arlington, TX, USA
taylor.johnson@gmail.com

Abstract

Safety verification of hybrid dynamical systems relies crucially on the ability to reason about reachable sets of continuous systems whose evolution is governed by a system of ordinary differential equations (ODEs). Verification tools are often restricted to handling a particular class of continuous systems, such as e.g. differential equations with constant right-hand sides, or systems of affine ODEs. More recently, verification tools capable of working with *non-linear* differential equations have been developed. The behavior of non-linear systems is known to be in general extremely difficult to analyze because solutions are rarely available in closed-form. In order to assess the practical utility of the various verification tools working with non-linear ODEs it is very useful to maintain a set of verification problems. Similar efforts have been successful in other communities, such as automated theorem proving, SAT solving and numerical analysis, and have accelerated improvements in the tools and their underlying algorithms. We present a set of 65 safety verification problems featuring non-linear polynomial ODEs and for which we have proofs of safety. We discuss the various issues associated with benchmarking the currently available verification tools using these problems.

1 Introduction

For verifying safety properties of hybrid systems, it is crucial to have the means of reasoning about safety properties of purely continuous systems that determine state evolution inside the *operating modes*.

In computer science, emphasis has traditionally been placed on working with hybrid systems in which the continuous modes are governed by relatively simple ODEs. For instance, safety verification of systems with ODEs possessing constant right-hand sides and right-hand sides bounded within real intervals is aided by the fact that reachable sets of such continuous systems can be computed exactly. Progress has been made on verifying safety in systems with linear and affine continuous dynamics (with tools such as PHAVer [13] and SpaceEx [14]). This is a much more difficult problem, since reachable sets of linear ODEs cannot in general be phrased in a decidable theory, which is only known to be possible for some special classes of systems [22, 16, 18].

It is a well-known fact that non-linear ODEs can exhibit behaviour that is impossible under affine or linear dynamics [19]. Their expressive power allows for modelling very rich dynamic phenomena, but comes at the price of making the reachability analysis much more difficult. A major obstacle is the fact that solutions to non-linear ODEs cannot in general be obtained as closed-form expressions, i.e. finite expressions in terms of polynomials and elementary functions such as $\exp, \sin, \cos, \ln$, etc. Hybrid systems with non-linear ODEs are not at all uncommon in control theory; this is especially true of the class of *piecewise-smooth systems* (sometimes called *variable structure systems*), which are used in the design of *sliding mode controllers* [10].

A number of tools and approaches have been developed that enable safety verification of non-linear systems (e.g. [7, 21, 36, 31, 30, 29, 24, 17, 15, 37]). The methods currently in existence differ in a number of aspects; for instance, the level of automation they provide, the generality of system and inputs specifications, etc. These important (and at times subtle) differences make the tools difficult to compare objectively. One approach to address the issue could be to push for a consensus in the community about a useful and fairly general class of systems of interest that we should all work on. However, any such enterprise would be necessarily artificial for the time being as there is no generally agreed-upon classification of differential equations. In this work, we rather advocate a pragmatic approach: that of creating a database of benchmarks that can be used for a comprehensive assessment of the existing and future verification tools. The hope would be to steer the research towards working with a growing set of examples that a variety of related communities care about. If such a set were available, a tool (or an approach) could easily be seen to be more powerful if it is able to handle (parse, verify, solve, etc.) a larger proportion of those examples. Determining which verification tool is "better" cannot be entirely objective as it would further need to take into account the tool's running time performance, memory requirements, level of automation, etc. However, we believe that the problem of comparing verification tools can, at least in part, be addressed by collecting verification benchmarks and converting them to a single standardized input format. While this effort is only a first step towards a more ambitious goal, we feel it is important to initiate the process of gathering interesting verification problems and making them available to the community.

Similar efforts have been successfully undertaken in fields such as automated theorem proving (e.g. the TPTP problem library [3]), SAT solving (where competitions, e.g. [1], have led to drastic improvements in the performance of SAT solvers in the last two decades) and numerical analysis [39], resulting in improved quality of the tools and their underlying algorithms.

Contributions

We (I) provide a set of 65 safety verification problems featuring non-linear systems, for all of which the safety property is known to hold. Further, we (II) discuss the current challenges in comparing verification tools working with non-linear continuous dynamics and (III) outline ideas for addressing some of these difficulties.

2 Benchmarks

We have collected a set of 65 safety verification problems featuring non-linear ODEs, which we have gathered from existing papers treating the problem of unbounded time safety verification [24, 9, 11, 37] and invariant generation for non-linear systems (e.g. [6]). The problems we have collected all share the property of having proofs of safety that were obtained using the methods presented in the pertinent papers (or having proofs that are immediate from the results described therein).

In general, in order to fully state a safety verification problem, one requires four pieces of information:

1. The system of ODEs, written using vector notation as $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$, where $f : \mathbb{R}^n \to \mathbb{R}^n$.
2. The mode invariant, denoted $H \subseteq \mathbb{R}^n$, which defines the region where the system may evolve along the solution to the system of ODEs.

3. The set of initial states $X_0 \subseteq \mathbb{R}^n$.
4. The set of unsafe (or forbidden) states $X_u \subseteq \mathbb{R}^n$.

Remark Note that it is sufficient to consider autonomous ODEs, i.e. those in which the right-hand side does not depend explicitly on the independent time variable t, because one may always augment the system with $\dot{t} = 1$ and treat t as a state variable. Furthermore, in many cases it is also sufficient to only consider polynomial problems because it is often possible to re-cast safety verification problems with non-polynomial terms to problems only featuring polynomial functions (see e.g. [25, 28]).

The problem is to show that it is impossible for the system to evolve into a forbidden state $\boldsymbol{x}_u \in X_u$ from any initial state $\boldsymbol{x}_0 \in X_0$ by following the solution $\varphi_t(\boldsymbol{x}_0)$ to the system of ODEs $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ for any time while it remains within the evolution constraint H. Formally, this may be written down as

$$\forall t \geq 0.\ \forall \boldsymbol{x}_0 \in X_0.\ (\forall \tau \in [0, t].\ \varphi_\tau(\boldsymbol{x}_0) \in H) \to \varphi_t(\boldsymbol{x}_0) \notin X_u.$$

In bounded-time safety verification one is only interested in showing safety up to some finite time bound $T \geq 0$, i.e.

$$\forall t \in [0, T].\ \forall \boldsymbol{x}_0 \in X_0.\ (\forall \tau \in [0, t].\ \varphi_\tau(\boldsymbol{x}_0) \in H) \to \varphi_t(\boldsymbol{x}_0) \notin X_u.$$

Clearly, if the safety property holds for unbounded time, it is guaranteed for any finite time bound, but not conversely. Since all the problems we have gathered are non-linear and have proofs of unbounded-time safety, we may designate this class of problems `NONLIN-UNBOUND-TIME-SAFE` in order to distinguish it from other classes of problems that we may wish to add later on, such as e.g. provably safe linear systems, or provably unsafe systems, etc. In this section we will illustrate some of the safety verification problems featuring 2-dimensional ODEs. The full set of the 65 problems is available from http://verivital.com/hyst/benchmark-nonlinear/

Example 2.1 (Non-linear example [9]). Dai et al. in [9] studied safety verification using barrier certificates, illustrating their approach using the following system:

$$\dot{x} = 2x - xy,$$
$$\dot{y} = 2x^2 - y.$$

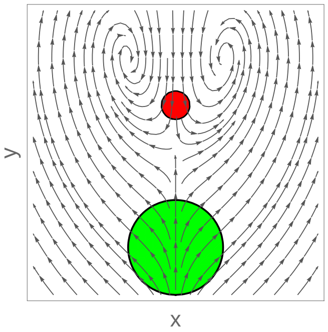


Figure 1: Non-linear system in the safety verification problem from [9].

The set of initial states is given by $x^2 + (y + 2)^2 \leq 1$ and the set of unsafe states is $x^2 + (y - 1)^2 \leq \frac{9}{100}$ (shown in green and red respectively in Fig. 1). The evolution constraint is taken to be the real plane $\mathbb{R}^2$.

Example 2.2 (FitzHugh-Nagumo system example [6])**.** Ben Sassi et al. [6] reported a method for generating polyhedral invariants for polynomial ODEs and applied it to the FitzHugh-Nagumo system:

$$\dot{x} = -\frac{x^3}{3} + x - y + \frac{7}{8},$$
$$\dot{y} = \frac{2}{25}\left(x - \frac{4y}{5} + \frac{7}{10}\right).$$

With the knowledge of the invariant, by considering initial states that lie inside the invariant,

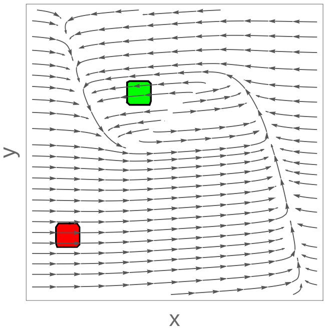


Figure 2: Safety verification in the FitzHugh-Nagumo system.

e.g. $-1 \leq x \leq -0.5 \ \land \ 1 \leq y \leq 1.5$ and letting $-2.5 \leq x \leq -2 \ \land \ -2 \leq y \leq -1.5$ represent the forbidden states, all of which lie entirely outside the invariant, one may conclude the safety property. Fig 2 shows the phase portrait along with the initial and the unsafe states (in green and red, respectively).

Example 2.3 ([37], ODE from [12], Ex. 10.15 (i))**.** In previous work [37], a non-linear ODE from a textbook on the qualitative theory of planar ODEs [12]

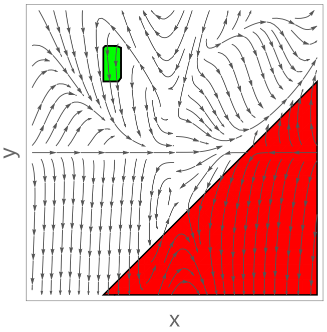


Figure 3: Safety verification problem from [37].

$$\dot{x} = -42x^7 + 68x^6y - 46x^5y + 258x^4y + 156x^3y + 50x^2y + 20xy^6 - 8y^7,$$
$$\dot{y} = y\left(1110x^6 - 220x^5y - 3182x^4y + 478x^3y^3 + 487x^2y^4 - 102xy^5 - 12y^6\right),$$

was used to create a safety verification problem where the initial states are given by $x > -1 \wedge x < -\frac{3}{4} \wedge y \leq \frac{3}{2} \wedge y \geq 1$ and the forbidden states satisfy the inequality $x > y + 1$ (shown respectively in green and red in Fig. 3).

2.1 Problem format

We have chosen to store our verification problems in a format used by the SpaceEx verification tool for hybrid systems [14]. While SpaceEx currently cannot work with non-linear differential equations, its input format is sufficiently simple and convenient. A given problem in this format is stored in two separate files

1. An `.xml` file storing the ODE $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ and the mode invariant H of the system.
2. A `.cfg` file detailing the initial set X_0 and the set of forbidden states X_u.

For example, the verification problem described in Example 2.2, may be stored in the two files shown in Fig. 4 and Fig. 5.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<sspaceex xmlns="http://www-verimag.imag.fr/xml-namespaces/sspaceex" version="0.2" math="SpaceEx">
 <component id="fitzhugh_nagumo_ben_sassi_girard_2">
 <param name="x" type="real" local="false" d1="1" d2="1" dynamics="any"/>
 <param name="y" type="real" local="false" d1="1" d2="1" dynamics="any"/>
 <location id="1" name="p">
  <invariant>true</invariant>
  <flow>x'==7/8+x-x^3/3-y &amp; y'==(2*(7/10+x-(4*y)/5))/25</flow>
 </location>
 </component>
</sspaceex>
```

Figure 4: FitzHugh-Nagumo system dynamics, illustrated in Fig. 2.

```
system = fitzhugh_nagumo_ben_sassi_girard_2
initially = "-1<=x & x<=-0.5 & 1<=y & y<=1.5"
forbidden = "-2.5<=x & x<=-2 & -2<=y & y<=-1.5"
output-variables = x,y
scenario = stc
directions = box
set-aggregation = "none"
sampling-time = 0.5
flowpipe-tolerance = 0.25
time-horizon = 9
iter-max = 4
output-format = GEN
verbosity = m
output-error = 0.001
rel-err = 1.0e-12
abs-err = 1.0e-15
```

Figure 5: SpaceEx configuration file specifying the initial and forbidden states.

3 Challenges

In using any significantly broad set of verification benchmarks, one faces a number of challenges if one wishes to use them to compare safety verification methods and tools. Firstly, in contrast to the world of SAT/SMT solving or automated theorem proving, verification of continuous systems has not matured to the point where the community has agreed upon an input standard that can be used to exchange problems (such as SMT-LIB [2] or TPTP [3]). Also, unlike with numerical analysis or simulation, general safety verification problems need not have point initial conditions, but rather a set of initial states that may be uncountably infinite, and not necessarily "nice" (e.g. may be disconnected, non-convex, unbounded, etc.). Below we outline some important challenges that stand in the way of benchmarking existing verification tools.

- Tools for bounded-time safety verification based on computing flowpipes enclosing reachable sets of non-linear ODEs, such as e.g. Flow*, are often limited in the nature of the initial and the forbidden sets of states. In particular, the underlying algorithms used in these tools require the set of initial states to be bounded (unlike in Example 2.3); ideally given by a hyper-rectangle (unlike Example 2.1). On the other hand, methods for automatic unbounded-time safety verification based on searching for appropriate continuous invariants (e.g. [29, 37]) are capable of working with much broader classes of initial and forbidden regions. For instance, semi-algebraic initial regions that are unbounded, non-convex, or whose description features a combination of conjunctions and disjunctions do not present a problem.

  Remark At the same time, tools based on flowpipe construction can sometimes give a sense of the "hardness" of the verification problem when they fail to prove safety up to some given time bound, whereas invariant-based verification tools typically do not provide useful insights into the nature or the difficulty of the problem when they fail.

- Tools that employ interval arithmetic often require bounds on the state variables of the system (e.g. HSolver [33, 34], dReach [21]), which technically renders them inapplicable to safety verification problems where the evolution constraint H is unbounded, e.g. given by $\mathbb{R}^n$.
- Certain tools (e.g. Flow*) cannot work with sets described by strict inequalities (such as the forbidden states in Example 2.3). While it would be sound to simply over-approximate the closure of such sets by relaxing the inequalities to be non-strict, this step currently needs to be performed manually by the user and (inevitably) affects the reachability analysis.
- The performance of tools often depends heavily on the user-specified options, such as e.g. the fixed/adaptive time steps used for the verified integration, error tolerances, etc. It is presently not apparent how one might automatically translate "good" settings from one verification tool to another, or indeed automatically arrive at good settings for a particular tool in the first place. Thus, some verification tools that are designed to be fully automatic rely crucially on the user choosing the right settings, which is typically difficult for a non-expert.
- Some unbounded-time verification methods (e.g. [31]) likewise require significant manual input from the user, such as e.g. selecting templates for polynomial functions. It is yet unclear how these methods can be meaningfully compared to methods that provide a greater level of automation.
- Uncertainty in the continuous dynamics is permitted by some verification tools (e.g. Flow*), but not others.

4 Outlook

Safety verification problems for non-linear systems are very useful for assessing the utility and efficiency of invariant generation methods (e.g. [35, 38, 6, 29, 26, 40, 17, 37]), as well as tools based on verified integration of ODEs (e.g [7, 27, 20, 21]). We are hopeful that maintaining and further populating the set of verification benchmarks will result in improvements to the existing capabilities offered by the tools for both bounded and unbounded-time safety verification. Improvements in invariant generation would also greatly benefit deductive verification tools for hybrid systems, such as theorem provers (e.g. [30, 15, 23]).

At least some of the challenges outlined in the previous section can potentially be addressed using HyST [5], a source transformation tool for hybrid systems that takes as input a hybrid system verification problem in the SpaceEx format and translates it into formats accepted by other verification tools. In addition to translating between the various problem formats, HyST is able to work with its internal representation of the verification problem through so-called *model transformation passes*, which can address issues that affect particular verification tools. For instance, currently HyST can add identity reset maps to transitions in hybrid automata, split transition guards with disjunctions, etc. A potentially interesting future transformation pass could be implemented in HyST to convert continuous systems with uncertainty into *hybrid* systems in which there is no uncertainty in the continuous dynamics, e.g. following the work of Ramdani et al. [32].

At present, HyST can translate problems into formats accepted by Flow*, dReach, HyCreate [4], HyComp [8] and SpaceEx. An interesting future direction would be to extend it to also work with invariant generation tools and add model transformation passes to soundly convert safety verification problems that currently cannot be processed by some of the verification tools into a form that is amenable to analysis.

In collecting safety verification benchmarks it is profitable to find a useful classification. One could separate verification problems for continuous systems into classes depending on certain features, such as:

- the type of continuous dynamics, e.g. constant/linear/non-linear,
- the dimensionality of the system (i.e. the number of state variables, $|\boldsymbol{x}|$),
- the type of safety verification (i.e. bounded versus unbounded time),
- the nature of the evolution constraint (e.g. bounded versus unbounded state space),
- the nature of the initial and forbidden set (bounded versus unbounded; if bounded, hyper-rectangles versus more general sets), and
- the nature of the verification problem itself (i.e. is the system safe or unsafe?).

Such a classification will certainly become important in the future as more verification problems are gathered and added to our collection. Our initial set of 65 problems (which we tentatively labelled `NONLIN-UNBOUND-TIME-SAFE`) belongs to one of the most general classes under this scheme, since it makes few assumptions about the nature of the verification problem. This generality makes it difficult to use the problems for benchmarking existing tools, but at the same time serves to bring out their current limitations.

Acknowledgements The authors would very much like to thank the anonymous reviewers for their careful reading, pertinent points of critique and valuable suggestions for improving this document. The material presented in this paper is based upon work supported by the National Science Foundation (NSF) under grant numbers CNS 1464311 and CCF 1527398, the Air Force Research Laboratory (AFRL) through contract number FA8750-15-1-0105, and the

Air Force Office of Scientific Research (AFOSR) under contract number FA9550-15-1-0258. The U.S. government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of AFRL, AFOSR, or NSF.

References

[1] The international SAT competitions web page. http://www.satcompetition.org/. Accessed: 2016-02-15.

[2] SMT-LIB the satisfiability modulo theories library. http://smtlib.cs.uiowa.edu/. Accessed: 2016-02-15.

[3] The TPTP problem library for automated theorem proving. http://www.cs.miami.edu/~tptp/. Accessed: 2016-02-15.

[4] Stanley Bak. HyCreate: A tool for overapproximating reachability of hybrid automata. http://stanleybak.com/projects/hycreate/hycreate.html. Accessed: 2016-02-15.

[5] Stanley Bak, Sergiy Bogomolov, and Taylor T. Johnson. HYST: a source transformation and translation tool for hybrid automaton models. In *HSCC*, pages 128–133. ACM, 2015.

[6] M.A. Ben Sassi, A. Girard, and S. Sankaranarayanan. Iterative computation of polyhedral invariants sets for polynomial dynamical systems. In *CDC*, pages 6348–6353, Dec 2014.

[7] Xin Chen, Erika Ábrahám, and Sriram Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *CAV*, pages 258–263, 2013.

[8] Alessandro Cimatti, Alberto Griggio, Sergio Mover, and Stefano Tonetta. HyComp: An SMT-based model checker for hybrid systems. In *TACAS*, pages 52–67, 2015.

[9] Liyun Dai, Ting Gan, Bican Xia, and Naijun Zhan. Barrier certificates revisited. *CoRR*, abs/1310.6481, 2013.

[10] Raymond A. DeCarlo, Stanisław H. Żak, and Gregory P. Matthews. Variable structure control of nonlinear multivariable systems: a tutorial. *Proceedings of the IEEE*, 76(3):212–232, 1988.

[11] A Djaballah, A. Chapoutot, M. Kieffer, and O Bouissou. Construction of Parametric Barrier Functions for Dynamical Systems using Interval Analysis. *ArXiv e-prints*, June 2015.

[12] Freddy Dumortier, Jaume Llibre, and Joan C. Artés. *Qualitative Theory of Planar Differential Systems*. Springer, 2006.

[13] Goran Frehse. PHAVer: Algorithmic verification of hybrid systems past HyTech. In *HSCC*, volume 3414, pages 258–273. Springer, 2005.

[14] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. SpaceEx: Scalable Verification of Hybrid Systems. In *CAV*, volume 6806 of *LNCS*. Springer, 2011.

[15] Nathan Fulton, Stefan Mitsch, Jan-David Quesel, Marcus Völp, and André Platzer. KeYmaera X: An axiomatic tactical theorem prover for hybrid systems. In *CADE*, volume 9195 of *LNCS*. Springer, 2015.

[16] Ting Gan, Mingshuai Chen, Liyun Dai, Bican Xia, and Naijun Zhan. Decidability of the reachability for a family of linear vector fields. In *ATVA*, pages 482–499, 2015.

[17] Khalil Ghorbal and André Platzer. Characterizing algebraic invariants by differential radical invariants. In *TACAS*, volume 8413, pages 279–294. Springer, 2014.

[18] Emmanuel Hainry. Reachability in linear dynamical systems. In *Logic and Theory of Algorithms, CiE 2008*, volume 5028 of *LNCS*, pages 241–250, 2008.

[19] Jack K. Hale and Joseph P. LaSalle. Differential equations: Linearity vs. nonlinearity. *SIAM Review*, 5(3):249–272, July 1963.

[20] Fabian Immler. Verified reachability analysis of continuous systems. In *TACAS*, pages 37–51, 2015.

[21] Soonho Kong, Sicun Gao, Wei Chen, and Edmund M. Clarke. dreach: δ-reachability analysis for hybrid systems. In *TACAS*, pages 200–205. Springer, 2015.

[22] Gerardo Lafferriere, George J. Pappas, and Sergio Yovine. Symbolic reachability computation for families of linear vector fields. *J. Symb. Comput.*, 32(3):231–253, 2001.

[23] Jiang Liu, Jidong Lv, Zhao Quan, Naijun Zhan, Hengjun Zhao, Chaochen Zhou, and Liang Zou. A calculus for hybrid CSP. In *Programming Languages and Systems*, volume 6461 of *LNCS*, pages 1–15. Springer, 2010.

[24] Jiang Liu, Naijun Zhan, and Hengjun Zhao. Computing semi-algebraic invariants for polynomial dynamical systems. In *EMSOFT*, pages 97–106, New York, NY, USA, 2011. ACM.

[25] Jiang Liu, Naijun Zhan, Hengjun Zhao, and Liang Zou. Abstraction of elementary hybrid systems by variable transformation. In *FM*, pages 360–377, 2015.

[26] Nadir Matringe, Arnaldo Vieira Moura, and Rachid Rebiha. Generating invariants for non-linear hybrid systems by linear algebraic methods. In *SAS*, volume 6337 of *LNCS*, pages 373–389. Springer, 2011.

[27] Nedialko S. Nedialkov. Interval Tools for ODEs and DAEs. In *SCAN*, pages 4–4, Sept 2006.

[28] André Platzer. Differential-algebraic dynamic logic for differential-algebraic programs. *J. Log. Comput.*, 20(1):309–352, 2010.

[29] André Platzer and Edmund M. Clarke. Computing differential invariants of hybrid systems as fixedpoints. In *CAV*, pages 176–189, 2008.

[30] André Platzer and Jan-David Quesel. KeYmaera: A hybrid theorem prover for hybrid systems. In *IJCAR*, volume 5195 of *LNCS*, pages 171–178. Springer, 2008.

[31] Stephen Prajna and Ali Jadbabaie. Safety verification of hybrid systems using barrier certificates. In *HSCC*, volume 2993 of *LNCS*, pages 477–492. Springer, 2004.

[32] N. Ramdani, N. Meslem, and Y. Candau. A hybrid bounding method for computing an over-approximation for the reachable set of uncertain nonlinear systems. *Automatic Control, IEEE Transactions on*, 54(10):2352–2364, Oct 2009.

[33] Stefan Ratschan and Zhikun She. HSolver. http://hsolver.sourceforge.net/, 2004. Accessed: 2016-02-15.

[34] Stefan Ratschan and Zhikun She. Safety verification of hybrid systems by constraint propagation-based abstraction refinement. *ACM Transactions on Embedded Computing Systems (TECS)*, 6(1):8, 2007.

[35] Sriram Sankaranarayanan. Automatic invariant generation for hybrid systems using ideal fixed points. In *HSCC*, pages 221–230, New York, NY, USA, 2010. ACM.

[36] B. I. Silva, Keith Richeson, Bruce Krogh, and Alongkrit Chutinan. Modeling and verifying hybrid dynamic systems using CheckMate. In *ADPM*, 2000.

[37] Andrew Sogokon, Khalil Ghorbal, Paul B. Jackson, and André Platzer. A method for invariant generation for polynomial continuous systems. In *VMCAI*, volume 9583 of *LNCS*. Springer, 2016.

[38] Ashish Tiwari. Generating box invariants. In *HSCC*, volume 4981 of *LNCS*, pages 658–661. Springer, 2008.

[39] Hoang-Dung Tran, Luan Viet Nguyen, and Taylor T. Johnson. Benchmark: A nonlinear reachability analysis test set from numerical analysis. In *Applied Verification for Continuous and Hybrid Systems Workshop*, Seattle, Washington, April 2015.

[40] Hengjun Zhao, Naijun Zhan, and Deepak Kapur. Synthesizing switching controllers for hybrid systems by generating invariants. In *Theories of Programming and Formal Methods*, pages 354–373, 2013.

Decoupling Abstractions of Non-linear Ordinary Differential Equations *

Andrew Sogokon[1], Khalil Ghorbal[2], and Taylor T. Johnson[1]

[1] Vanderbilt University, Nashville, TN, USA
{andrew.sogokon|taylor.johnson}@vanderbilt.edu
[2] INRIA, Rennes, Brittany, France
khalil.ghorbal@inria.fr

Abstract. We investigate decoupling abstractions, by which we seek to simulate (i.e. abstract) a given system of ordinary differential equations (ODEs) by another system that features completely independent (i.e. uncoupled) sub-systems, which can be considered as separate systems in their own right. Beyond a purely mathematical interest as a tool for the qualitative analysis of ODEs, decoupling can be applied to verification problems arising in the fields of control and hybrid systems. Existing verification technology often scales poorly with dimension. Thus, reducing a verification problem to a number of independent verification problems for systems of smaller dimension may enable one to prove properties that are otherwise seen as too difficult. We show an interesting correspondence between Darboux polynomials and decoupling simulating abstractions of systems of polynomial ODEs and give a constructive procedure for automatically computing the latter.

Keywords: ordinary differential equations, Darboux polynomials, simulation, abstraction, decoupling

1 Introduction

Simulation relations are an important concept in the study of both discrete and continuous dynamical systems. Informally speaking, a system simulates another system if it over-approximates its set of possible behaviours. In practice, when analyzing systems, one often wants to construct simulations of the original system that are in some sense “simpler” to analyze. Then, by demonstrating some property of interest in the simulation one may infer the property in the original system.

In [22] Sankaranarayanan investigated an interesting technique for constructing simulations of continuous systems by employing change of basis transformations. It was shown how *linearizing* change of basis transformations of non-linear

* This work was supported by the Air Force Research Laboratory (AFRL) through contract number FA8750-15-1-0105 and the Air Force Office of Scientific Research (AFOSR) under contract numbers FA9550-15-1-0258 and FA9550-16-1-0246.

systems of ODEs can yield simulations in which the dynamics is given by a system of linear ODEs. The motivation for considering such transformations is clear, since linear systems cannot exhibit some of the rich dynamic phenomena found in their non-linear counterparts and are more amenable to analysis [11]. In this paper we consider simulations of non-linear ODEs of a different kind: instead of linear dynamics, we seek to construct simulations that are potentially non-linear, but whose analysis can be performed in a lower-dimensional space than that of the original system.

Although our focus in this paper is on analyzing purely continuous systems, the methods we present are motivated by the broader goal of aiding the task of automatic verification of *hybrid dynamical systems* whose continuous modes are governed by non-linear ODEs. Hybrid systems combine discrete and continuous behaviour; their formal modelling and verification is of increasing interest and importance to modern engineering, where discrete digital controllers are used to control continuously evolving physical plants. In recent years, verification technology for hybrid systems has seen significant advances and a number of interesting case studies have been reported, e.g. verification of train control systems [20,29], aircraft collision avoidance protocols [13,1], descent guidance control software in a lunar lander [28] and satellite rendezvous manoeuvres [14], to give a few examples. However, non-linear ODEs appearing in hybrid system models often present a serious challenge to verification due to their inherent complexity. In this paper we seek to overcome some aspects of this hurdle by constructing simulations of non-linear ODEs with structure that more readily lends itself to analysis.

1.1 Contributions

In this paper we `(I)` define *decoupled simulating abstractions* of non-linear ODEs, discuss their utility and relationship to *first integrals* [10] and *constant-scale continuous consecutions* [23]. `(II)` We give an algorithm for checking whether a given set of polynomial *abstract basis functions* can be used to create a decoupled abstraction of a system of polynomial ODEs and then `(III)` employ the theory of *Darboux polynomials* [10] to give sufficient criteria for non-existence of polynomial abstract basis functions suitable for constructing decoupled polynomial abstractions. Lastly, `(IV)` we show how Darboux polynomials can be used to construct the abstract basis functions for decoupled abstractions whenever they exist. We conclude with a summary of our findings, an overview of related work and directions for future research.

1.2 Preliminaries

An autonomous n-dimensional system of ODEs has the following form:

$$\begin{aligned} \dot{x}_1 &= f_1(x_1, x_2, \ldots, x_n), \\ &\vdots \\ \dot{x}_n &= f_n(x_1, x_2, \ldots, x_n), \end{aligned}$$

where for $i \in \{1, \dots, n\}$ each $f_i : \mathbb{R}^n \to \mathbb{R}$ is a real-valued function (typically C^1), and $\dot{x}_i$ denotes the time derivative of x_i, i.e. $\frac{d}{dt}x_i(t)$. In applications, constraints are often imposed on the states where the system is allowed to evolve, i.e. the system may only evolve inside some given set $H \subseteq \mathbb{R}^n$, which is known as the *evolution constraint*. We may write this more concisely using vector notation as $\dot{\boldsymbol{x}} = f(\boldsymbol{x}),\ \boldsymbol{x} \in H$. Here $\dot{\boldsymbol{x}} = (\dot{x}_1, \dots, \dot{x}_n)$ and $f : \mathbb{R}^n \to \mathbb{R}^n$ is a *vector field* generated by the system, i.e. $f(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \dots, f_n(\boldsymbol{x}))$ for all $\boldsymbol{x} \in \mathbb{R}^n$. When no evolution constraint is specified, H is assumed to be $\mathbb{R}^n$.

A *solution* to the initial value problem for the system of ODEs $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ with initial value $\boldsymbol{x}_0 \in \mathbb{R}^n$ is a differentiable function $\varphi_t(\boldsymbol{x}_0) : (a, b) \to \mathbb{R}^n$ defined for all t within some non-empty extended real interval including zero, i.e. $t \in (a, b) \subseteq \mathbb{R} \cup \{\infty, -\infty\}$, where $a < 0 < b$, and such that $\frac{d}{dt}\varphi_t(\boldsymbol{x}_0) = f(\varphi_t(\boldsymbol{x}_0))$ for all $t \in (a, b)$. If the solution $\varphi_t(\boldsymbol{x}_0)$ is available in closed-form,[3] then one can answer questions about the temporal behaviour of the system (such as e.g. safety and liveness) by analyzing the closed-form expression. In practice, however, it has long been established that explicit closed-form solutions to non-linear ODEs are highly uncommon [11].

In this paper we will work with systems of ODEs whose right-hand sides are given by polynomials in the state variables $x_1, \dots, x_n$. Formally, we say that $f_i \in \mathbb{R}[X_1, \dots, X_n]$ for all $i \in \{1, \dots, n\}$, where $\mathbb{R}[X_1, \dots, X_n]$ denotes the ring of multivariate polynomials with real coefficients and indeterminates $X_1, \dots, X_n$. We write $f_i(x_1, \dots, x_n)$ when we wish to make it clear that the polynomial is treated as a function, with indeterminates replaced by the appropriate variables. Polynomial systems of ODEs are necessarily locally Lipschitz continuous, which guarantees existence of unique solutions on some non-trivial time interval for any initial value $\boldsymbol{x}_0 \in \mathbb{R}^n$ (by the Picard-Lindelöf theorem; see e.g. [27]).

1.3 Coupling

Given a system of ODEs $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$, the *maximum coupling coefficient* (henceforth mcc) is the size of the largest sub-system with no independent sub-systems. To define rigorously, we construct a finite *coupling graph* $CG = (V, E)$, where the set of vertices is precisely the set of state variables, i.e. $V = \{x_1, \dots, x_n\}$, and there is an edge from x_i to some other vertex x_j, i.e. $(x_i, x_j) \in E$ with $i \neq j$, if and only if $\frac{\partial f_i}{\partial x_j} \neq 0$. The *coupling coefficients* cc are a finite multiset of natural numbers corresponding to the orders (i.e. the numbers of vertices) of all the weakly connected components in CG. The coefficient mcc is defined to be the maximum order of the weakly connected components in CG, i.e. $\mathsf{mcc} \equiv \max \mathsf{cc}$.

Definition 1 (Uncoupled system). *A system of ODEs $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ is* uncoupled *if and only if its $\mathsf{mcc} = 1$, i.e. if the rate of change of each state variable is completely independent of the other variables.*[4]

[3] By this we understand a *finite* expression in terms of polynomials and elementary functions such as $\sin, \cos, \exp, \ln$, etc.

[4] An equivalent (but less flexible) definition would state that a system $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ is uncoupled if and only if the Jacobian matrix J_f is diagonal.

Example 1. Consider the following two planar polynomial systems:

$$\begin{aligned} \dot{x}_1 &= x_1^2 x_2 + 5x_1 - 1, & \qquad \dot{x}_1 &= x_1^3 + 5x_1 - 10, \\ \dot{x}_2 &= 3x_2^3 + 2x_1 x_2 - x_1. & \qquad \dot{x}_2 &= 2x_2^2 + 3x_2 + 1. \end{aligned}$$

The system on the left has $\mathsf{mcc} = 2$ because the vertices $\{x_1, x_2\}$ in the coupling graph have edges connecting them in both directions, since $\frac{\partial}{\partial x_2}(x_1^2 x_2 + 5x_1 - 1) = x_1^2 \neq 0$ and $\frac{\partial}{\partial x_1}(3x_2^3 + 2x_1 x_2 - x_1) = 2x_2 - 1 \neq 0$. On the other hand, the system on the right has $\mathsf{mcc} = 1$ (i.e. is uncoupled) because $\frac{\partial}{\partial x_2}(x_1^3 + 5x_1 - 10) = 0$ and $\frac{\partial}{\partial x_1}(2x_2^2 + 3x_2 + 1) = 0$ and therefore the vertices $\{x_1, x_2\}$ in the graph are disconnected.

Uncoupled systems are appealing first and foremost because their 1-dimensional sub-systems can be analyzed independently, following a standard technique for 1-dimensional flows (see e.g. [25, Chapter 2]). For instance, consider the 1-dimensional system $\dot{x} = x^3 + 5x^2 + x - 10$. This system evolves on the real line and has fixed points at the real roots of $x^3 + 5x^2 + x - 10$, of which there are three: $\{-2, \frac{1}{2}\left(-3 - \sqrt{29}\right), \frac{1}{2}\left(-3 + \sqrt{29}\right)\}$. The direction of the flow is to the right whenever the graph of $\dot{x}$ is above zero (i.e. the rate of change of x is positive) and to the left when it is below (the rate of change is negative), as shown in Figure 1.3.

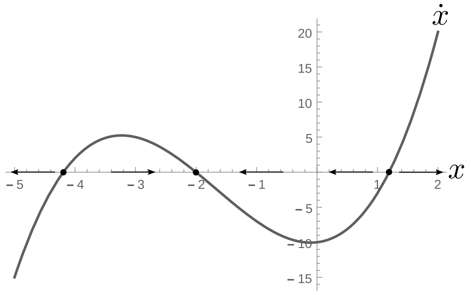


Fig. 1. Analysis of the 1-dimensional system $\dot{x} = x^3 + 5x^2 + x - 10$.

From inspecting the figure, one can readily see how one can construct the set of reachable states of any given initial point x_0 in a 1-dimensional polynomial system $\dot{x} = f(x)$: either the point is a root of the right-hand side, i.e. $f(x_0) = 0$, in which case x_0 remains invariant and the reachable set is simply $\{x_0\}$, or x_0 is not a root, i.e. $f(x) \neq 0$, in which case the reachable set is an interval of the form $[x_0, r)$ or $(r, x_0]$, where $r \in \mathbb{R} \cup \{\infty, -\infty\}$ is either a real root of f or it is ∞ or $-\infty$, respectively (if there are no real roots in the direction of motion). The reachable set from any initial point $\boldsymbol{x}_0 \in \mathbb{R}^n$ in a uncoupled system can thus also be bounded by combining the independent reachable sets in the 1-dimensional sub-systems.

Bounded-time reachable set computation using *verified integration* methods is also made easier because large systems of non-linear ODEs are typically expensive to integrate using methods that yield tight enclosures [16] (such as *Taylor models* [3,17]), whereas in an uncoupled system, no matter how large, each 1-dimensional sub-system can be integrated separately. An enclosure of the solution to the whole system at some time t can then be constructed directly from the enclosures of the solutions to the sub-systems at that time.

2 Decoupled Simulating Abstractions

In what follows, we will adopt the approach described by Sankaranarayanan in [22] to define *simulating abstractions* of non-linear ODEs using appropriate change of basis transformations.

Definition 2 (Simulating abstraction). *For a system $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$, $\boldsymbol{x} \in H$, where $f : \mathbb{R}^n \to \mathbb{R}^n$ is locally Lipschitz continuous, equipped with an initial set of states $X_0 \subseteq \mathbb{R}^n$, a system $\dot{\boldsymbol{\alpha}} = G(\boldsymbol{\alpha})$, $\boldsymbol{\alpha} \in \widehat{H}$, where $G : \mathbb{R}^m \to \mathbb{R}^m$ is locally Lipschitz continuous and equipped with an initial set of states $\widehat{X}_0 \subseteq \mathbb{R}^m$ is a* simulating abstraction *if there exists a smooth (i.e. C^∞) mapping $\boldsymbol{\alpha} : \mathbb{R}^n \to \mathbb{R}^m$ such that: (i) $\boldsymbol{\alpha}(X_0) \subseteq \widehat{X}_0$, (ii) $\boldsymbol{\alpha}(H) \subseteq \widehat{H}$, and (iii) for any trajectory (i.e. solution in non-negative time) $\varphi_\tau(\boldsymbol{x}_0) : [0, T) \to H$ of the system $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$, $\boldsymbol{x} \in H$, the trajectory $\boldsymbol{\alpha} \circ \varphi_\tau(\boldsymbol{x}_0) : [0, T) \to \widehat{H}$ is a trajectory of $\dot{\boldsymbol{\alpha}} = G(\boldsymbol{\alpha})$, $\boldsymbol{\alpha} \in \widehat{H}$.*

To ensure that the last condition in the above definition holds, it is sufficient to show that $G(\boldsymbol{\alpha}(\boldsymbol{x})) = J_{\boldsymbol{\alpha}} \cdot f(\boldsymbol{x})$, where $J_{\boldsymbol{\alpha}}$ is the Jacobian of the smooth mapping $\boldsymbol{\alpha}$ w.r.t. the state variables $x_1, \ldots, x_n$ (see [22, Theorem 2.1]), i.e.

$$G(\boldsymbol{\alpha}) = \begin{pmatrix} \frac{\partial \alpha_1}{\partial x_1} & \cdots & \frac{\partial \alpha_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \alpha_m}{\partial x_1} & \cdots & \frac{\partial \alpha_m}{\partial x_n} \end{pmatrix} \cdot \begin{pmatrix} f_1 \\ \vdots \\ f_n \end{pmatrix}.$$

Definition 3 (Lie derivative). *For a given system of ODEs $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$, the* Lie derivative *of a smooth function $p : \mathbb{R}^n \to \mathbb{R}$ is given by*

$$\mathfrak{L}_f(p) = \nabla p \cdot f = \sum_{i=1}^{n} \frac{\partial p}{\partial x_i} \cdot f_i.$$

Note that since $f_i(\boldsymbol{x}) = \frac{dx_i}{dt}$, $\mathfrak{L}_f(p) = \left(\sum_{i=1}^n \frac{\partial p}{\partial x_i} \cdot \frac{dx_i}{dt}\right) = \frac{dp}{dt}$ i.e. the total derivative of the function p with respect to time, which we denote by $\dot{p}$.

Let us recall that the gradient ∇p gives the vector of all the partial derivatives of p, i.e. $\nabla p \equiv \left(\frac{\partial p}{\partial x_1}, \frac{\partial p}{\partial x_2}, \ldots, \frac{\partial p}{\partial x_n}\right)$, and thus the necessary condition for (iii) in Definition 2 to be satisfied may be equivalently stated as:

$$G(\boldsymbol{\alpha}) = \begin{pmatrix} \nabla \alpha_1 \\ \vdots \\ \nabla \alpha_m \end{pmatrix} \cdot f = \begin{pmatrix} \mathfrak{L}_f(\alpha_1) \\ \vdots \\ \mathfrak{L}_f(\alpha_m) \end{pmatrix}.$$

Remark 1. It is important to note that, following Def. 2, solutions to simulating abstractions are guaranteed to exist for at least as long as they do in the concrete system. This property is crucial to soundness of the abstraction. A rather different, but in a certain sense more general, concept was explored by Platzer, who introduced *differential ghosts* [19], where the original dynamics is augmented by introducing some fresh variables whose rate of change may feature the newly introduced variables themselves, but is not restricted in the same way as in Def. 2. However, extra care needs to be taken to ensure that the solutions of the newly defined dynamics exist for at least as long as the solutions to the original system (e.g. see [19, Proof of Theorem 38]).

Definition 4 (Decoupling simulating abstraction). *Given a system of ODEs $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$, a simulating abstraction $\dot{\boldsymbol{\alpha}} = G(\boldsymbol{\alpha})$ is* decoupling *if and only if the equalities $\mathfrak{L}_f(\alpha_1) = G_1(\alpha_1), \ldots, \mathfrak{L}_f(\alpha_m) = G_m(\alpha_m)$ hold, where $(G_1, \ldots, G_m) = G$. Such an abstraction is thus uncoupled:*

$$\begin{aligned} \dot{\alpha}_1 &= G_1(\alpha_1), \\ &\vdots \\ \dot{\alpha}_m &= G_m(\alpha_m). \end{aligned}$$

In what follows, we will give some examples of how first integrals (see e.g. [10]) and constant-scale continuous consecutions [23] provide the abstract basis functions $\boldsymbol{\alpha}$ which lead to decoupling simulating abstractions.

Example 2 (Algebraically integrable system). The 3-dimensional system

$$\begin{aligned} \dot{x}_1 &= x_1(x_3 - x_2), \\ \dot{x}_2 &= x_2(x_1 - x_3), \\ \dot{x}_3 &= x_3(x_2 - x_1), \end{aligned}$$

has two independent polynomial conserved quantities, i.e. first integrals, given by $\alpha_1 = x_1x_2x_3$ and $\alpha_2 = x_1 + x_2 + x_3$ (see [8, Ex. 75]). If we let $\boldsymbol{\alpha} = (\alpha_1, \alpha_2)$, we obtain the decoupling simulating abstraction $\dot{\boldsymbol{\alpha}} = \mathbf{0}$, i.e. $\dot{\alpha}_1 = 0, \dot{\alpha}_2 = 0$.

Remark 2. A polynomial system $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ of size n is *algebraically integrable* if it possesses $n-1$ independent polynomial conserved quantities (also known as first integrals; see [10,8]), i.e. polynomials $\{\alpha_1, \ldots, \alpha_{n-1}\}$, where for all $i = 1, \ldots, n-1$ one has $\mathfrak{L}_f(\alpha_i) \equiv \dot{\alpha}_i = 0$. Algebraic integrability is a very powerful property, since it allows one to construct tight approximations of the *orbit* $\gamma(\boldsymbol{x}_0)$, i.e. the reachable set from $\boldsymbol{x}_0 \in \mathbb{R}^n$ in positive as well as negative time. That is, for any given point $\boldsymbol{x}_0 \in \mathbb{R}^n$, if one evaluates each first integral $\alpha_1, \ldots, \alpha_{n-1}$ at $\boldsymbol{x}_0$, one obtains real constants $c_1, \ldots, c_{n-1}$. The orbit through $\boldsymbol{x}_0$ is guaranteed to satisfy the formula $\alpha_1 = c_1 \wedge \cdots \wedge \alpha_{n-1} = c_{n-1}$, which corresponds to a (real) algebraic subset of $\mathbb{R}^n$ given by the common real roots of the polynomials $\alpha_i - c_i$. Every point $\boldsymbol{\alpha}_0 \in \mathbb{R}^{n-1}$ in such an abstract system $\dot{\boldsymbol{\alpha}} = \mathbf{0}$ is invariant and corresponds to a real (and invariant) algebraic set containing the orbit of the system $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$.

Polynomials p such that $\mathfrak{L}_f(p) = \lambda p$ for some $\lambda \in \mathbb{R}$ generalize polynomial first integrals[5] and were investigated by Sankaranarayanan et al. in [23], where they were used in *constant-scale continuous consecution* conditions. In general, if one can find polynomials $\alpha_1, \ldots, \alpha_m$ that satisfy $\mathfrak{L}_f(\alpha_i) = \lambda_i \alpha_i$, $\lambda_i \in \mathbb{R}$ for all $i \in \{1, \ldots, m\}$, then one obtains a decoupling abstraction of the form

$$\begin{aligned} \dot{\alpha}_1 &= \lambda_1 \alpha_1, \\ &\vdots \\ \dot{\alpha}_m &= \lambda_m \alpha_m. \end{aligned}$$

We generalize this idea to decoupling polynomial abstractions by considering polynomial functions $\alpha_i \in \mathbb{R}[X_1, \ldots, X_n]$ such that $\mathfrak{L}_f(\alpha_i) = G_i(\alpha)$, where $G_i \in \mathbb{R}[X]$, i.e. the derivative of α_i may be expressed as a polynomial in α_i with real coefficients.

Example 3 (Decoupling simulating abstraction). Consider the coupled system:

$$\begin{aligned} \dot{x}_1 &= \frac{1}{3}(1 - 3x_1 + 2x_1^2 - 6x_2 + 4x_1x_2 + 2x_2^2), \\ \dot{x}_2 &= \frac{1}{3}(-1 - 3x_1 + x_1^2 + 2x_1x_2 + x_2^2)\,. \end{aligned}$$

Let $\alpha_1 = x_1 + x_2 - 1$, $\alpha_2 = x_1 - 2x_2$. If we consider $\boldsymbol{\alpha} = (\alpha_1, \alpha_2)$, we arrive at the following system (left), which can be expressed as an uncoupled system in the new basis (right):

$$\begin{aligned} \dot{\alpha}_1 &= -2x_1 + x_1^2 - 2x_2 + 2x_1x_2 + x_2^2, & \dot{\alpha}_1 &= \alpha_1^2 - 1, \\ \dot{\alpha}_2 &= 1 + x_1 - 2x_2, & \dot{\alpha}_2 &= \alpha_2 + 1\,. \end{aligned}$$

3 Existence and Generation of Abstraction Polynomials

In what follows, we investigate the existence of polynomials that can be used to construct decoupling simulating abstractions of a given system. We show in Section 3.1 that their existence (to a given polynomial degree) is decidable and give a sufficient criterion for their non-existence (to a given degree) based on the existence of so-called *Darboux polynomials* (e.g. see [10]). We then explore the problems of *checking* and *generation*. The checking problem is concerned with determining whether a given candidate polynomial is suitable for constructing a decoupling simulating abstraction. In Section 3.2 we describe a procedure for solving the checking problem. In Section 3.3 we present a technique for generating all suitable polynomials for the decoupling abstract basis (up to a given polynomial degree).

[5] i.e. p is a first integral if $\mathfrak{L}_f(p) = \lambda p$ where $\lambda = 0$.

3.1 Decidability and Darboux Existence Criterion

For polynomial systems of ODEs $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$, the problem of finding a non-constant polynomial in the state variables, $p \in \mathbb{R}[X_1, \dots, X_n]$, for the decoupling abstract basis reduces to searching for those p such that $\mathfrak{L}_f(p) = G(p)$, where $G \in \mathbb{R}[X]$, i.e. G is a univariate polynomial with real coefficients. There may, however, be no such polynomial. Fortunately, it is decidable to check for existence of such a p.

Proposition 1 (Existence of decoupling abstract basis polynomials). *Given a positive integer d and a polynomial system $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$, it is decidable to check whether there exists a polynomial $p \in \mathbb{R}[X_1, \dots, X_n]$ of total degree d such that $\mathfrak{L}_f(p) = G(p)$, where $G \in \mathbb{R}[X]$ is a univariate polynomial with real coefficients.*

Proof. The problem can be stated as a sentence in the theory of real arithmetic which is decidable [26]. Let $\lambda_0, \dots, \lambda_k$ denote the unknown coefficients of the generic polynomial template p of degree d, where $k := \binom{n+d}{d} - 1$ is the number of *non-constant* monomials of degree at most d in n variables. The Lie derivative $\mathfrak{L}_f(p)$ can therefore be symbolically computed (Def. 3). Let $\kappa_0, \dots, \kappa_m$ denote the unknown coefficients of the polynomial $G \in \mathbb{R}[X]$ where $m := \lceil \deg(\mathfrak{L}_f(p))/d \rceil$. The decision problem stated in the proposition is therefore equivalent to deciding the truth of the following sentence:

$$\begin{aligned}&\exists\, (\lambda_0, \dots, \lambda_k) \in \mathbb{R}^{k+1}.\ \exists\, (\kappa_0, \dots, \kappa_m) \in \mathbb{R}^{m+1}. \\ &\qquad \forall (X_1, \dots, X_n) \in \mathbb{R}^n.\ d > 0 \wedge \mathfrak{L}_f(p) - (\kappa_0 + \kappa_1 p + \dots + \kappa_m p^m) = 0\ .\end{aligned}$$

If λ_0 denotes the constant term of the generic polynomial template p, then the condition $d > 0$ is equivalent (over the reals) to the inequality $\sum_{0<i\leq k} \lambda_i^2 > 0$, ensuring that p is non-constant. □

In practice, there is currently no question of applying existing decision procedures to formulas constructed in the proof or Prop. 1. The complexity of the most popular procedure for real quantifier elimination (CAD, due to Collins [4]) is doubly exponential in the number of variables. In Section 3.3 we will pursue a more promising method of searching for decoupling abstract basis polynomials. First, we shall recall so-called *Darboux polynomials*, a well-known tool in the study integrability of dynamical systems (e.g. see [10]), and use them to give a non-existence criterion for decoupling abstract basis polynomials. We then explore an interesting relationship between the two concepts.

Definition 5 (Darboux polynomial). *A polynomial $q \in K[X_1, \dots, X_n]$, where K is a field of characteristic zero (e.g. $\mathbb{C}, \mathbb{R}, \mathbb{Q}$), is a* Darboux polynomial[6] *for $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ iff $\mathfrak{L}_f(q) = \lambda q$, for some $\lambda \in K[X_1, \dots, X_n]$.*

[6] When q is a constant, the Darboux polynomial is *trivial*[10, Definition 2.14]. In this paper we will generally be interested in the non-trivial case.

Proposition 2 (Criterion for non-existence of decoupled abstractions). *If a given system $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ does not admit any Darboux polynomials over $\mathbb{C}$ of degree d, then there is no polynomial $p \in \mathbb{R}[X_1, \ldots, X_n]$ of degree d such that $\mathfrak{L}_f(p) = G(p)$ for some* non-constant $G \in \mathbb{R}[X]$.

Proof. We prove the contrapositive. Suppose there exists a polynomial $p \in \mathbb{R}[X_1, \ldots, X_n]$ such that $\mathfrak{L}_f(p) = G(p)$, where $G \in \mathbb{R}[X]$ is non-constant. By the fundamental theorem of algebra, G must have at least one complex root $c \in \mathbb{C}$. Therefore $G = (X - c)H$, where $H \in \mathbb{C}[X]$. We see that $(p - c)$ is a Darboux polynomial for the system because

$$\mathfrak{L}_f(p - c) = \mathfrak{L}_f(p) - \mathfrak{L}_f(c) = \mathfrak{L}_f(p) = G(p) = (p - c)H(p).$$

The degree of the Darboux polynomial $p - c$ is equal to the degree of p. □

3.2 Checking Abstraction Polynomial Candidates

Before proceeding to methods for generating decoupling abstract basis polynomials for polynomial systems $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$, we discuss the (easier) problem of checking if for a given $p \in \mathbb{R}[X_1, \ldots, X_n]$ one can write $\mathfrak{L}_f(p) = G(p)$, where $G \in \mathbb{R}[X]$.

In general, given any two polynomials $P, p \in \mathbb{R}[X_1, \ldots, X_n]$, if $\deg(P) \geq \deg(p)$, one may obtain a rewriting $P = G(p)$ by solving a system of linear equations. One proceeds by first defining the maximum degree of a possible G to be $d = \lceil \deg(P)/\deg(p) \rceil$. If an appropriate rewriting exists, then there is guaranteed to be a solution $(\lambda_0, \ldots, \lambda_d) \in \mathbb{R}^{d+1}$ to the equation $P = \lambda_0 + \lambda_1 p + \lambda_2 p^2 + \cdots + \lambda_d p^d$. By expanding and equating the monomial coefficients on both sides one arrives at a system of linear equations (of size no larger than the number of monomials of P) in the real variables $\lambda_0, \ldots, \lambda_d$. Thus, in the worst case, one has to solve a linear system with $d+1$ variables and $\binom{n+\deg(P)}{\deg(P)}$ equational constraints. A solution may be computed using a linear solver and the rewriting polynomial constructed as $G = \lambda_0 + \lambda_1 X + \cdots + \lambda_d X^d$. In what follows, we will refer to the procedure for obtaining the rewriting as Rewrite, that is Rewrite(P, p) gives G whenever $P = G(p)$.

Remark 3. It is worth remarking that the procedure Rewrite can be implemented by performing successive *polynomial reductions*, rather than by solving a linear program. Polynomial reduction extends polynomial division for univariate polynomials to the multivariate case and in general requires the computation of *Gröbner bases*. This functionality is available in most modern computer algebra systems.

3.3 Automated Generation of Decoupling Abstractions

A highly efficient method for synthesizing polynomial first integrals for polynomial ODEs was reported by Matringe et al. in [15], where the synthesis problem is reduced to computing the null space of a matrix with real entries. In [7], the authors extended the work of Matringe et al. to generate real algebraic invariants of polynomial ODEs, giving a search procedure for the most general class of invariant sets that can be expressed using polynomial equations. The same procedure can be used to generate Darboux polynomials over the reals or over the complexes only by changing the underlying computational field. In general, there is no known bound for the degree of Darboux polynomials in a given system. However, the automatic generation procedure is guaranteed to find all the independent Darboux polynomials for the system up to a given degree.

In this section, we explore the relationship between polynomials in a decoupling abstract basis and Darboux polynomials. This relationship will enable us to exploit the efficient symbolic generation methods reported in [15,7]. We outline a procedure for constructing polynomials p such that $\mathfrak{L}_f(p) = G(p)$, where $G \in \mathbb{R}[X]$, from a list of automatically generated Darboux polynomials (up to some given degree). The procedure will require two lemmas given below.

We note first that whenever q is a Darboux polynomial, any constant multiple of q, i.e. aq for some $a \in \mathbb{R}$ or $\mathbb{C}$, is also Darboux. A similar property holds for the decoupling abstract basis functions in simulating abstractions.

Lemma 1. *If $p \in \mathbb{R}[X_1, \ldots, X_n]$ is such that $\mathfrak{L}_f(p) = G(p)$ where $G \in \mathbb{R}[X]$, then $s = ap + b$ for any real numbers a, b, is such that $\mathfrak{L}_f(s) = F(s)$, where $F \in \mathbb{R}[X]$.*

Proof. If $a = 0$ then $\mathfrak{L}_f(s) = \mathfrak{L}_f(b) = 0$ and F is simply the zero polynomial in $\mathbb{R}[X]$. If $a \neq 0$, by our hypothesis we have $\mathfrak{L}_f(p) = G(p)$. Let us write $p = \frac{s-b}{a}$ and note that

$$\mathfrak{L}_f(s) = \mathfrak{L}_f(ap + b) = a\mathfrak{L}_f(p) + \mathfrak{L}_f(b) = a\mathfrak{L}_f(p) = aG(p) = aG\left(\frac{s-b}{a}\right).$$

We see that $\mathfrak{L}_f(s) = aG\left(\frac{s-b}{a}\right)$ is a polynomial in s with real coefficients. □

One consequence of Lem. 1 is that whenever we assume the existence of a polynomial p such that $\mathfrak{L}_f(p) = G(p)$ for some $G \in \mathbb{R}[X]$, it always suffices to assume the existence of a decoupling abstract basis polynomial $p - r$ for any real number r.

In Prop. 2 we established that the existence of decoupling abstract basis polynomials p is related to the existence of a special Darboux polynomial $p - c$ for some complex number c. For any polynomial s, we denote by s^* the polynomial obtained by setting the constant term of s to zero. For instance, if $s = x + 1$ then, $s^* = x$. Thus, for the (Darboux) polynomial $p - c$, one has $(p - c)^* = p^*$ (by definition of the * operator) and therefore p^* is a decoupling abstract basis polynomial by Lem. 1, since it is an offset of the polynomial p by a real number (the constant term of p). Therefore, if one generates Darboux polynomials over the complex numbers and finds a Darboux polynomial q such that q^* is a polynomial over the reals (i.e. all the coefficients of q^* are real numbers), then q^* is potentially a decoupling abstract basis polynomial, which can be checked by solving a linear program, i.e. by running Rewrite($\mathfrak{L}_f(q^*), q^*$), as outlined in Section 3.2.

Nevertheless, generating Darboux polynomials over the complex numbers will not necessarily return Darboux polynomials q such that q^* is a polynomial over the reals even if the latter exist. For instance, if $q = x^2 + xy + c$ is a Darboux polynomial with some complex constant term c, then the procedure may return $\imath q$ instead of q ($\imath$ being the imaginary number satisfying $\imath^2 = -1$), although we are rather interested in looking for q. Enforcing such a constraint in the procedure for generating Darboux polynomials will require solving mixed non-linear equations where some variables are real and some are complex numbers. To avoid solving mixed problems, we can easily adapt the generation procedure to produce *monic* Darboux polynomials for any variable ordering, for instance the lexicographic order $X_1 > \cdots > X_n$. Recall that monic univariate polynomials are those polynomials where the leading coefficient (i.e. the coefficient of the leading monomial) is equal to 1. In the multivariate case, the notion of leading coefficient additionally requires a monomial ordering. For instance, for the order $X_1 > X_2$, the leading monomial of the polynomial $2X_1X_2 + X_1^2$ is X_1^2 and therefore the leading coefficient is 1, whereas the leading monomial in the reverse lexicographic ordering $X_2 > X_1$ is X_1X_2 and the leading coefficient is 2.

Lemma 2. *Given a polynomial $q \in \mathbb{C}[X_1, \dots, X_n]$, let $p \in \mathbb{C}[X_1, \dots, X_n]$ be the monic polynomial $\frac{q}{\mathsf{LC}(q)}$, where $\mathsf{LC}(q)$ is the* leading coefficient *of q with respect to some fixed monomial ordering. There exists a non-zero complex number z such that $(zq)^* \in \mathbb{R}[X_1, \dots, X_n]$ if and only if $p^* \in \mathbb{R}[X_1, \dots, X_n]$.*

Proof. Suppose there exists such a non-zero complex number z such that $(zq)^* \in \mathbb{R}[X_1, \dots, X_n]$. Since $z\mathsf{LC}(q) = \mathsf{LC}(zq)$ we have that $\frac{zq}{\mathsf{LC}(zq)} = \frac{zq}{z\mathsf{LC}(q)} = \frac{q}{\mathsf{LC}(q)} = p$, therefore $\frac{1}{\mathsf{LC}(zq)}(zq) = p$ and $\frac{1}{\mathsf{LC}(zq)}(zq)^* = p^*$. Since $\mathsf{LC}(zq) \in \mathbb{R}$, we have $p^* \in \mathbb{R}[X_1, \dots, X_n]$. Conversely, if $p^* \in \mathbb{R}[X_1, \dots, X_n]$, take $z = \frac{1}{\mathsf{LC}(q)}$ so that $(zq)^* = p^*$. □

We now describe a procedure for generating decoupling abstract basis polynomials. Suppose we are given all the independent Darboux polynomials in $\mathbb{C}[X_1, \dots, X_n]$ for the system $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ up to some degree $d > 0$. By Prop. 2, if there exists a polynomial $p \in \mathbb{R}[X_1, \dots, X_n]$ of degree $d' \leq d$ such that $\mathfrak{L}_f(p) = G(p)$, where $G \in \mathbb{R}[X]$ is non-constant, then there necessarily exists a Darboux polynomial q of degree d' such that q^* is a polynomial over the reals, i.e. $q^* \in \mathbb{R}[X_1, \dots, X_n]$. This fact suggests a simple search method. Below we describe the three main steps in the procedure.

1. For a fixed positive integer d, automatically generate all monic Darboux polynomials for the system up to degree d with coefficients in $\mathbb{C}$.
2. For each generated Darboux polynomial q check if $q^* \in \mathbb{R}[X_1, \dots, X_n]$ and if so, store q^* as a candidate in a list L.
3. For all polynomials q^* in L, run Rewrite($\mathfrak{L}_f(q^*), q^*$). If q^* is a decoupling abstract basis polynomial, the rewriting procedure will return $G \in \mathbb{R}[X]$ s.t. $\mathfrak{L}_f(q^*) = G(q^*)$.

Example 4. Consider the following system

$$\dot{x}_1 = \frac{1}{3}(1 + x_1 - 2x_2 + 2(1 + (-1 + x_1 + x_2)^2))$$
$$\dot{x}_2 = \frac{1}{3}(-x_1 + 2x_2 + (-1 + x_1 + x_2)^2)$$

The automatic generation procedure for Darboux polynomials over $\mathbb{C}$ up to degree 1 gives us $(q_1, q_2, q_3) = (1+x_1-2x_2, (-1+\imath)+x_1+x_2, (-1-\imath)+x_1+x_2)$. In this case, q_1^*,q_2^* and q_3^* are all candidates for the short list L. Since $q_2^* = q_3^*$, $L = \{x_1 - 2x_2, x_1 + x_2\}$. Running $\textsc{Rewrite}(\mathfrak{L}_f(q_1^*), q_1^*)$ and $\textsc{Rewrite}(\mathfrak{L}_f(q_2^*), q_2^*)$ returns $2 - 2X + X^2$ and $1 + X$, respectively. Thus, letting $(\alpha_1, \alpha_2) = (q_1^*, q_2^*)$, we obtain the decoupled abstraction:

$$\dot{\alpha}_1 = 2 - 2\alpha_1 + \alpha_1^2,$$
$$\dot{\alpha}_2 = 1 + \alpha_2.$$

In general, a Darboux polynomial q, with $q^* \in \mathbb{R}[X_1, \ldots, X_n]$, is not necessarily a decoupling abstract basis polynomial. For instance, in the system $\dot{x}_1 = x_1x_2, \dot{x}_2 = x_2$, one has x_1 as a Darboux polynomial; however x_1 is not a decoupling abstract basis polynomial because $\mathfrak{L}_f(x_1) = x_1x_2$ cannot be rewritten as polynomial in x_1 only. The checking procedure $\textsc{Rewrite}(\mathfrak{L}_f(x_1), x_1)$ will thus fail to produce a solution.

It is natural to ask under what extra conditions is a Darboux polynomial q satisfying $q^* \in \mathbb{R}[X_1, \ldots, X_n]$ also a decoupling abstract basis polynomial. The following theorem explores this connection.

Theorem 1. *Given a system of polynomial ODEs $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$, there exists a polynomial $p \in \mathbb{R}[X_1, \ldots, X_n]$ such that $\mathfrak{L}_f(p) = G(p)$, where $G \in \mathbb{R}[X]$ is of degree $d > 0$, if and only if the system has d Darboux polynomials $q_1, \ldots, q_d \in \mathbb{C}[X_1, \ldots, X_n]$ satisfying:*

(i) $q_1^* = q_2^* = \cdots = q_d^* \in \mathbb{R}[X_1, \ldots, X_n]$,
(ii) $\mathfrak{L}_f(q_1) = \mathfrak{L}_f(q_2) = \cdots = \mathfrak{L}_f(q_d) = rq_1q_2\cdots q_d$, $r \in \mathbb{R}$,
(iii) *for all $i = 1, \ldots, d$, either $q_i^* - q_i \in \mathbb{R}$ or there exists $j \neq i$, $j = 1, \ldots, d$, such that $q_i = \bar{q}_j$.*

Proof. Suppose there exists a $p \in \mathbb{R}[X_1, \ldots, X_n]$ such that $\mathfrak{L}_f(p) = G(p)$. When $G \in \mathbb{R}[X]$ is a non-constant polynomial of degree d, it can be factorized as $r(X - c_1)\cdots(X - c_d)$, where $r \in \mathbb{R}$ and the roots c_i are either real numbers, or complex numbers that come in conjugate pairs, i.e. if $c_i \in \mathbb{C}$ is a root of G, then its complex conjugate $\bar{c}_i$ is also a root. In the proof of Prop. 2 we have seen that for any such factor $(X - c_i)$ the polynomial $q_i = p - c_i$ is a Darboux polynomial for the system such that $\mathfrak{L}_f(q_i) = G(p)$. The properties (i), (ii) and (iii) follow immediately.

Conversely, let us assume that there are d Darboux polynomials $q_1, q_2, \ldots, q_d$ satisfying properties (i), (ii) and (iii). Then for any $r \in \mathbb{R}$ we have

$$rq_1q_2\cdots q_d = r(q_1^* - c_1)(q_2^* - c_2)\cdots(q_d^* - c_d),$$

where each $c_i = q_i^* - q_i$ is, by definition, a constant. By property (i) we have $q_1^* = q_2^* = \cdots = q_d^* \in \mathbb{R}[X_1, \ldots, X_n]$, so let us take $p = q_1^* = q_2^* = \cdots = q_d^*$ to obtain $r(q_1^* - c_1)(q_2^* - c_2) \cdots (q_d^* - c_d) = r(p - c_1)(p - c_2) \cdots (p - c_d)$. One can now write this as $r(p - c_1)(p - c_2) \cdots (p - c_d) = G(p)$, where $G \in \mathbb{R}[X]$ has degree d. The coefficients of G are real because by (iii) the roots c_i come in complex conjugate pairs. Since $q_i = q_i^* - (q_i^* - q_i) = p - c_i$, we have $\mathfrak{L}_f(q_i) = \mathfrak{L}_f(p - c_i) = \mathfrak{L}_f(p) - \mathfrak{L}_f(c_i) = \mathfrak{L}_f(p)$ and by (ii) $\mathfrak{L}_f(p) = r(p - c_1)(p - c_2) \cdots (p - c_d) = G(p)$. □

Notice that Rewrite does not require all of the d Darboux polynomials in order to construct G. If a family of Darboux polynomials $\{q_1, \ldots, q_d\}$ as stated in Theorem 1 exists, it suffices to supply only one element, say q_1^*, to Rewrite, which will then find a rewriting of $\mathfrak{L}_f(q_1^*)$ as $G(q_1^*)$, with $G \in \mathbb{R}[X]$. If however, the algorithm fails, then the polynomial supplied was not obtained from such a family of Darboux polynomials and therefore cannot be used to obtain a rewriting of its derivative in terms of itself.

Theorem 1 exposes the structure inherent in systems for which one can find decoupled simulating abstractions. The requirements (i)–(iii) are indeed quite strong. Observe that when G is a linear polynomial with a real coefficient λ, i.e. is of the form $G(X) = \lambda X$ and therefore necessarily has one real root, Theorem 1 reduces to the conditions for constant-scale consecution [23].

Remark 4. Theorem 1 relies on generating Darboux polynomials in order to compute a decoupling abstraction of a given system of polynomial ODEs. Nevertheless, polynomials having *constant* Lie derivatives (that is, those p s.t. $\mathfrak{L}_f(p) = G(p)$ where G has degree zero) can also be used for decoupling abstractions, but are not covered by Theorem 1, which requires the degree of G to be positive. The special case when G has degree zero is also related to Darboux polynomials as follows: (i) when G is the zero polynomial, then the system has a first integral which is a special Darboux polynomial as discussed in Section 2, (ii) when G is a non-zero constant, then the augmented system $(\dot{\boldsymbol{x}}, \dot{t}) = (f(\boldsymbol{x}), 1)$ obtained by appending the time derivative to the original system has a polynomial first integral. More precisely, when $p \in \mathbb{R}[X_1, \ldots, X_n]$ and the $\mathfrak{L}_f(p)$ is a real constant, say r, then in the augmented system $\mathfrak{L}_{(f,1)}(p - rt) = \mathfrak{L}_{(f,1)}(p) - r = \mathfrak{L}_f(p) - r = 0$ and $p - rt$ is thus a polynomial first integral of the augmented system. One may thus handle this case by computing first integrals (e.g. using the approach described in [15]) before searching for more sophisticated decoupling polynomials where G has a positive degree.

4 Outlook

Verification problems for systems of ODEs can be soundly translated to verification problems for their simulating abstractions. Below we sketch the case of a standard safety verification problem (S_x, f, F_x), where one wishes to prove that a given property, encoded as the region $F_x \subset \mathbb{R}^n$, is always satisfied if the system $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ is initialised in $\boldsymbol{x}_0 \in S_x \subset \mathbb{R}^n$. If a decoupling abstraction $\dot{\boldsymbol{\alpha}} = G(\boldsymbol{\alpha})$

exists, one can attempt to solve the simpler *abstract* safety verification problem (S_y, G, F_y) where $(y_1, \dots, y_m) = (\alpha_1(\boldsymbol{x}), \dots, \alpha_m(\boldsymbol{x}))$, denoted henceforth by $\boldsymbol{y} = \boldsymbol{\alpha}(\boldsymbol{x})$, i.e. $\dot{\boldsymbol{y}} = G(\boldsymbol{y})$ is a decoupled simulating abstraction. The initial set in the new abstract coordinates, $S_y \subset \mathbb{R}^m$ (resp. F_y), is computed as a projection of the semialgebraic set $S_x \wedge \boldsymbol{y} = \boldsymbol{\alpha}(\boldsymbol{x})$, which is a subset of $\mathbb{R}^{n+m}$ (resp. $F_x \wedge \boldsymbol{y} = \boldsymbol{\alpha}(\boldsymbol{x})$), onto $\mathbb{R}^m$. Such a projection can in principle be obtained by eliminating the existential quantifiers in the following sentence

$$\exists\ (x_1, \dots, x_n) \in \mathbb{R}^n.\, S_x \ \wedge\ y_1 = \alpha_1(x_1, \dots, x_n) \ \wedge\ \cdots \ \wedge\ y_m = \alpha_m(x_1, \dots, x_n)\,.$$

The soundness of such an abstraction relies essentially on two facts: (i) the sets S_y and F_y are the exact images through $\boldsymbol{\alpha}$ of the sets S_x and F_x respectively (although using over-approximations of these sets is also sound) and (ii) the invariant regions of the decoupled abstract system, when expressed in terms of the old coordinates, define invariant regions of the original system (i.e. the abstraction is indeed *sound* [22, Theorem 2.2]). This means that if the safety problem holds true in the decoupled abstraction it also holds true in the original concrete system. If not, however, the abstraction may be too coarse.

Interesting directions for refining the abstraction include searching for more general simulating abstractions that are not necessarily completely decoupling. For instance, it is conceivable that a simulating abstraction may possess independent sub-systems that are of the form

$$\begin{aligned}\dot{\alpha}_i &= G_i(\alpha_i, \alpha_j),\\ \dot{\alpha}_j &= G_j(\alpha_i, \alpha_j),\end{aligned}$$

where $G_i, G_j \in \mathbb{R}[X_1, X_2]$ and $\alpha_i, \alpha_j \in \mathbb{R}[X_1, \dots, X_n]$ are the abstract basis functions. This idea is similar to the so-called algebraizing transformations, briefly discussed in [22, Definition 2.4]. The analysis of 2-dimensional (i.e. planar) polynomial ODEs is however vastly more difficult than the 1-dimensional case. Indeed, qualitative analysis of planar polynomial flows is an active area of mathematical research (e.g. see [6,5]). However, one hope is this greater generality would make simulating abstractions of this form more “common” in systems that one might encounter in applications.

Decoupling can help overcome some of the scalability issues in existing verification methodologies. For instance, in reachability analysis, *relational abstraction* [24] seeks to abstract the flow of a differential equation by an over-approximation of the reachability relation on the states of the system. Mathematically, a (timeless) relational abstraction of an autonomous system $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$ is a relation $R \subseteq \mathbb{R}^n \times \mathbb{R}^n$ such that $(\boldsymbol{x}, \boldsymbol{y}) \in R$ if $\boldsymbol{y}$ is reachable from $\boldsymbol{x}$ in finite time by following the flow of the system [24, Definition 4], i.e. if $\exists\, t \geq 0.\ \varphi_t(\boldsymbol{x}) = \boldsymbol{y}$. Computing timeless relational abstractions for non-linear systems is difficult because it reduces to searching for positive invariants in the extended system of ODEs $\dot{\boldsymbol{y}} = f(\boldsymbol{y}), \dot{\boldsymbol{x}} = \boldsymbol{0}$ with dimension $2n$, i.e. with twice the number of state variables [24, Definition 5, Lemma 1]. When the system is uncoupled, one can instead work with n extended systems $\dot{y}_i = f_i(y_i), \dot{x}_i = 0$, $i = 1, \dots, n$, each of dimension 2.

5 Related Work

Our work is closest in spirit to that of Sankaranarayanan [22], which studied simulating abstractions resulting from linearizing change of basis transformations. Our approach instead focused on simulating abstractions obtained via decoupling change of basis transformations.

Change of basis transformations are a standard technique for decoupling linear homogeneous systems of ODEs with constant coefficients, i.e. systems of the form $\dot{\boldsymbol{x}} = A\boldsymbol{x}$, where A is an $n \times n$ real matrix. A common technique applies when the matrix A has n real distinct eigenvalues and produces a decoupled linear homogeneous system $\dot{\boldsymbol{\alpha}} = B\boldsymbol{\alpha}$ of the *same dimension*, where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$ is made up of *linear functions* $\alpha_i : \mathbb{R}^n \to \mathbb{R}$ in the state variables $x_1, \ldots, x_n$ (see e.g. [21, §28.2, §28.3]); in particular, such a decoupling is always possible when A is a real symmetric matrix. In our work, we consider more general polynomial systems of ODEs and a more general class of polynomials to act as the new basis; additionally, we do not require the dimension of the resulting decoupled system to match that of the original system of coupled ODEs. In short, our focus is not placed on solving the system, but rather on automatically discovering simulating abstractions that are more amenable to analysis.

Girard and Pappas explored *approximate bisimulation* of continuous systems in [9], and Pappas earlier developed (exact) bisimulations between continuous linear systems [18]. However, these works employ a different notion of simulation and do not seek to make the structure of the simulation easier to analyze in the way that we do with decoupling, and are in practice limited to linear ODEs due to reliance on solving linear matrix inequalities (LMIs). Han and Krogh have also explored sound order reduction techniques for verification with reachability analysis, but their approach is also limited to linear ODEs [12]. In contrast to all these existing works that employ different techniques as well as different formal development, our decoupled simulating abstractions are applicable to non-linear polynomial ODEs, and as such, are developed using significantly different methods.

6 Conclusion

In this paper we explored a technique for constructing decoupling simulating abstractions of non-linear polynomial ODEs, which can be more easily analyzed because their 1-dimensional sub-systems may be treated independently. We employed the theory of Darboux polynomials to give a sufficient criterion for non-existence of decoupled simulating abstractions (up to a some maximum degree of the abstract basis polynomials; see Prop. 2). Lastly, we described how automatically generated Darboux polynomials (up to some given polynomial degree) can be used to construct abstract basis polynomials that can yield decoupling simulating abstractions. The abstractions developed in this paper are in essence a form of model transformation, which can be integrated in source transformation and translation tools such as HyST [2]; we leave this for future work.

Acknowledgements The authors would like to thank the anonymous reviewers for their careful reading and judicious critique and extend their thanks to Dr. André Platzer at Carnegie Mellon University for his technical questions and helpful insights into differential ghosts.

Bibliography

1. Abrial, J., Su, W., Zhu, H.: Formalizing hybrid systems with event-b. In: ASM. LNCS, vol. 7316, pp. 178–193. Springer (2012)
2. Bak, S., Bogomolov, S., Johnson, T.T.: HYST: a source transformation and translation tool for hybrid automaton models. In: HSCC. pp. 128–133. ACM (2015)
3. Berz, M., Makino, K.: Verified integration of odes and flows using differential algebraic methods on high-order taylor models. Reliable Computing 4(4), 361–369 (1998)
4. Collins, G.E.: Hauptvortrag: Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In: Automata Theory and Formal Languages, 2nd GI Conference, Kaiserslautern, May 20-23, 1975. LNCS, vol. 33, pp. 134–183. Springer (1975)
5. Conti, R., Galeotti, M.: Dynamical Systems, chap. Totally Bounded Cubic Systems in $\mathbb{R}^2$, pp. 103–171. Springer (2003)
6. Dumortier, F., Llibre, J., Artés, J.C.: Qualitative Theory of Planar Differential Systems. Springer (2006)
7. Ghorbal, K., Platzer, A.: Characterizing algebraic invariants by differential radical invariants. In: TACAS. LNCS, vol. 8413, pp. 279–294. Springer (2014)
8. Ginoux, J.M.: Differential Geometry Applied to Dynamical Systems, World Scientific Series on Nonlinear Science, vol. 66. World Scientific (2009)
9. Girard, A., Pappas, G.J.: Approximate bisimulation: A bridge between computer science and control theory. Eur. J. Control 17(5-6), 568–578 (2011)
10. Goriely, A.: Integrability and Nonintegrability of Dynamical Systems. Advanced series in nonlinear dynamics, World Scientific (2001)
11. Hale, J.K., LaSalle, J.P.: Differential equations: Linearity vs. nonlinearity. SIAM Review 5(3), 249–272 (Jul 1963)
12. Han, Z., Krogh, B.: Reachability analysis of hybrid control systems using reduced-order models. In: American Control Conference, 2004. Proceedings of the 2004. vol. 2, pp. 1183–1189 vol.2 (June 2004)
13. Jeannin, J., Ghorbal, K., Kouskoulas, Y., Gardner, R., Schmidt, A., Zawadzki, E., Platzer, A.: A formally verified hybrid system for the next-generation airborne collision avoidance system. In: TACAS. LNCS, vol. 9035, pp. 21–36. Springer (2015)
14. Johnson, T.T., Green, J., Mitra, S., Dudley, R., Erwin, R.S.: Satellite rendezvous and conjunction avoidance: Case studies in verification of nonlinear hybrid systems. In: Giannakopoulou, D., Méry, D. (eds.) FM 2012: Formal Methods - 18th International Symposium, Paris, France, August 27-31, 2012. Proceedings. LNCS, vol. 7436, pp. 252–266. Springer (2012)
15. Matringe, N., Moura, A.V., Rebiha, R.: Generating invariants for non-linear hybrid systems by linear algebraic methods. In: SAS. LNCS, vol. 6337, pp. 373–389. Springer (2010)
16. Nedialkov, N.S.: Interval Tools for ODEs and DAEs. In: 12th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics (SCAN). pp. 4–4 (Sept 2006)

17. Neher, M., Jackson, K.R., Nedialkov, N.S.: On taylor model based integration of odes. SIAM J. Numerical Analysis 45(1), 236–262 (2007)
18. Pappas, G.J.: Bisimilar linear systems. Automatica 39(12), 2035–2047 (2003)
19. Platzer, A.: A complete uniform substitution calculus for differential dynamic logic. Journal of Automated Reasoning pp. 1–47 (2016)
20. Platzer, A., Clarke, E.M.: Formal verification of curved flight collision avoidance maneuvers: A case study. In: FM. LNCS, vol. 5850, pp. 547–562. Springer (2009)
21. Robinson, J.C.: An introduction to ordinary differential equations. Cambridge University Press (2004)
22. Sankaranarayanan, S.: Change-of-bases abstractions for non-linear hybrid systems. Nonlinear Analysis: Hybrid Systems 19, 107 – 133 (2016)
23. Sankaranarayanan, S., Sipma, H.B., Manna, Z.: Constructing invariants for hybrid systems. Formal Methods in System Design 32(1), 25–55 (2008)
24. Sankaranarayanan, S., Tiwari, A.: Relational abstractions for continuous and hybrid systems. In: CAV. LNCS, vol. 6806, pp. 686–702. Springer (2011)
25. Strogatz, S.H.: Nonlinear Dynamics and Chaos. Westview Press (1994)
26. Tarski, A.: A decision method for elementary algebra and geometry. Bulletin of the American Mathematical Society 59 (1951)
27. Teschl, G.: Ordinary Differential Equations and Dynamical Systems, Graduate Studies in Mathematics, vol. 140. American Mathematical Society (2012)
28. Zhao, H., Yang, M., Zhan, N., Gu, B., Zou, L., Chen, Y.: Formal verification of a descent guidance control program of a lunar lander. In: FM. LNCS, vol. 8442, pp. 733–748. Springer (2014)
29. Zou, L., Lv, J., Wang, S., Zhan, N., Tang, T., Yuan, L., Liu, Y.: Verifying chinese train control system under a combined scenario by theorem proving. In: VSTTE. LNCS, vol. 8164, pp. 262–280. Springer (2013)

Operational models of piecewise-smooth systems [*]

Andrew Sogokon
Carnegie Mellon University
Pittsburgh, PA, USA
asogokon@cs.cmu.edu

Khalil Ghorbal
INRIA
Rennes, France
khalil.ghorbal@inria.fr

Taylor T. Johnson
Vanderbilt University
Nashville, TN, USA
taylor.johnson@vanderbilt.edu

ABSTRACT

This paper studies ways of constructing meaningful operational models of piecewise-smooth systems (PWS). The systems we consider are described by polynomial vector fields defined on non-overlapping semi-algebraic sets, which form a partition of the state space. Our approach is to give meaning to motion in systems of this type by automatically synthesizing operational models in the form of hybrid automata (HA). Despite appearances, it is in practice often difficult to arrive at satisfactory HA models of PWS. The different ways of building operational models that we explore in our approach can be thought of as defining different semantics for the underlying PWS. These differences have a number of interesting nuances related to phenomena such as chattering, non-determinism, so-called mythical modes and sliding behaviour.

Keywords

piecewise-smooth systems, hybrid automata, operational models, discontinuous differential equations.

1. INTRODUCTION

Many processes in which smooth continuous motion can be interrupted by discrete events can be represented by ordinary differential equations (ODEs) with discontinuities. As such, they are part of a broader class of dynamical systems, known as hybrid (also cyber-physical) systems, which combine discrete and continuous behaviour under a unified framework[1]. Hybrid systems are increasingly used in modelling and analyzing the behaviour of modern control systems employing embedded devices.

Systems described by discontinuous ODEs are sometimes referred to as *piecewise-smooth systems* (PWS). Their representation has proved popular in the control systems community because it provides a concise and convenient notation. However, a discontinuous system of ODEs explicitly only conveys information about the continuous dynamics of the system, along with a set of regions where state evolution is smooth; the discrete transition behaviour of the system between these regions is not explicitly elaborated.

There exist a number of specification formalisms, such as *hybrid automata* [1] and *hybrid programs* [34], whose semantics is clearly defined and which can serve as operational models for hybrid systems. Hybrid automata in particular have become very popular in the verification community. In a hybrid automaton, the discrete transition behaviour of the hybrid system is specified explicitly, which can often make these automata large and unwieldy even when specifying hybrid systems of relatively modest size. As a specification formalism, discontinuous ODEs provide a much more concise and manageable description of piecewise-smooth systems, albeit leaving many important details about their behaviour implicit.

Researchers working in computer science and control systems tend to put different emphasis on the importance of formal modelling and tend to use significantly different methods to model and reason about systems. One particular aspect of these differences is manifest in the temptation to treat hybrid automata naïvely as being merely syntactic variants of discontinuous ODEs when modelling piecewise-smooth systems. Subscribing to this view is, however, rather dangerous and can lead to unintended behaviour in the resulting models.

In this paper we study the challenges presented by the problem of transforming concise descriptions of piecewise-smooth systems in the form of discontinuous ODEs into formal operational models in the form of hybrid automata. Transformations that result in satisfactory models are, as we shall see, far from trivial to both formulate and effect. We develop automatic procedures for transforming piecewise-smooth systems with polynomial dynamics and semi-algebraic constraints into hybrid automata.

A number of different interpretations of the operational meaning of piecewise-smooth systems are possible, creating a degree of ambiguity about their intended behaviour (i.e. their semantics); this gives rise to significant differences in the form and the behaviour of the hybrid automata that one

[*] This work was supported by the Air Force Research Laboratory (AFRL) through contract number FA8750-15-1-0105 and the Air Force Office of Scientific Research (AFOSR) under contract numbers FA9550-15-1-0258 and FA9550-16-1-0246.

[1] Indeed, some of the earliest research in hybrid systems, e.g. in the work of Witsenhausen [44], began by considering precisely the systems where there are no "jumps" in the continuous state, but abrupt changes in the *dynamics* are possible.

EMSOFT'17 Seoul, Korea

ACM ISBN .

DOI: 10.1145/3126506

can construct. A number of important choices can be exercised in transforming PWS to HA in order to ensure that the resulting operational models reflect the desired interpretation.

1.1 Contributions

In this paper we describe a method for automatically constructing hybrid automata from descriptions of piecewise-smooth polynomial systems and thus build their operational models. We discuss aspects of the semantics of transitions directly related to phenomena such as chattering, non-determinism and the presence of so-called *mythical modes* in the underlying systems. We illustrate how our technique can be applied to model systems with so-called *sliding modes.* We conclude with a discussion of related work and an outlook for future research.

2. MATHEMATICAL PRELIMINARIES

2.1 Continuous systems and vector fields

A general n-dimensional autonomous system of first-order ODEs has the form:

$$\begin{aligned} \dot{x}_1 &= f_1(x_1, x_2, \dots, x_n), \\ &\vdots \\ \dot{x}_n &= f_n(x_1, x_2, \dots, x_n), \end{aligned}$$

where $f_i : \mathbb{R}^n \to \mathbb{R}$ are real-valued functions (typically C^1) for each $i = 1, \dots, n$ and $\dot{x}_i$ denotes the derivative of x_i with respect to time, i.e. $\frac{dx_i}{dt}$. Such a system defines a *vector field* $\mathfrak{f} : \mathbb{R}^n \to \mathbb{R}^n$, where $\mathfrak{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \dots, f_n(\boldsymbol{x}))$ for any $\boldsymbol{x} \in \mathbb{R}^n$. We will denote autonomous systems of ODEs using the more concise vector field notation, i.e. by writing $\dot{\boldsymbol{x}} = \mathfrak{f}(\boldsymbol{x})$.

In applications, it is often the case that the state of the system is required to only evolve within some prescribed set of "legal" states $M \subseteq \mathbb{R}^n$, which is known as the *mode invariant*, or *evolution constraint*. We will express this requirement concisely by writing $\dot{\boldsymbol{x}} = \mathfrak{f}(\boldsymbol{x}),\ \boldsymbol{x} \in M$. When no evolution constraint is specified, M is assumed to be $\mathbb{R}^n$.

A *solution* to the initial value problem for the system of ODEs $\dot{\boldsymbol{x}} = \mathfrak{f}(\boldsymbol{x})$ with initial value $\boldsymbol{x}_0 \in \mathbb{R}^n$ is a differentiable function $\boldsymbol{x} : (a, b) \to \mathbb{R}^n$, where $\boldsymbol{x}(t)$ is defined for all t within some non-empty extended real interval including zero, i.e. $t \in (a, b) \subseteq \mathbb{R} \cup \{\infty, -\infty\}$ where $a < 0 < b$, and such that $\boldsymbol{x}(0) = \boldsymbol{x}_0$ and $\frac{d}{dt}\boldsymbol{x}(t) = \mathfrak{f}(\boldsymbol{x}(t))$ for all $t \in (a, b)$. In what follows, we will denote the solution $\boldsymbol{x}(t)$ by writing $\varphi_t(\boldsymbol{x}_0)$, to emphasize the initial value. If the function $\varphi_t(\boldsymbol{x}_0)$ is available in closed-form[2], one can analyze the temporal behaviour of the system initialized in the state $\boldsymbol{x}_0$ by analyzing the closed-form expression. In practice, however, it has long been established that explicit closed-form solutions to non-linear ODEs are highly uncommon [20].

Systems of ODEs whose right-hand sides are *locally Lipschitz continuous* (e.g. polynomial functions fall under this class) guarantee existence of unique solutions on some non-trivial time interval (a, b) for any initial value $\boldsymbol{x}_0 \in \mathbb{R}^n$ (by the Cauchy-Lipschitz/Picard-Lindelöf theorem; see e.g. [39]).

2.2 Piecewise-smooth vector fields

Given a partition of some set $M \subseteq \mathbb{R}^n$ into finitely many non-overlapping subsets $M_1, \dots, M_m$, we consider a finite family of vector fields $\mathfrak{f}_i : \mathbb{R}^n \to \mathbb{R}^n$, where $i \in \{1, \dots, m\}$. By assigning the vector field $\mathfrak{f}_i$ from this family to the set M_i for each $i = 1, \dots, m$, we arrive at a vector field $\mathfrak{F} : M \to \mathbb{R}^n$ which is defined piecewise on M, i.e.

$$\mathfrak{F}(\boldsymbol{x}) = \begin{cases} \mathfrak{f}_1(\boldsymbol{x}) & \boldsymbol{x} \in M_1\,, \\ \vdots & \\ \mathfrak{f}_m(\boldsymbol{x}) & \boldsymbol{x} \in M_m\,. \end{cases} \tag{1}$$

At this point, let us remark that while the sets $M_1, \dots, M_m$ need not be differentiable manifolds, the corresponding vector fields $\mathfrak{f}_1, \dots, \mathfrak{f}_m$ are defined on $\mathbb{R}^n$. It is therefore meaningful to speak about motion occurring within the manifold $\mathbb{R}^n$ according to the systems of ODEs $\dot{\boldsymbol{x}} = \mathfrak{f}_i(\boldsymbol{x})$, but confined to the states within M_i. With this intuition, the vector field $\mathfrak{F}$ can be interpreted as describing a system of ODEs $\dot{\boldsymbol{x}} = \mathfrak{F}(\boldsymbol{x})$ with a piecewise-defined (and potentially discontinuous) right-hand side, i.e. explicitly given by

$$\dot{\boldsymbol{x}} = \mathfrak{F}(\boldsymbol{x}) \tag{2}$$

To precisely describe the motion taking place (within the set M) in such a system, in general one may no longer call upon the classical notion of solution developed for continuous ODEs.[3] Indeed, there is no single universally agreed-upon definition of solution for systems of ODEs with discontinuities. Extensions of the classical notion, such as Carathéodory solutions, among others [19], have been suggested, but these differ in the way they model certain dynamic behaviours and therefore give different meaning (i.e. semantics) to systems. An excellent accessible survey of discontinuous ODEs and the various *generalized solution* concepts developed for them was given by Cortés in [8]. Intuitively, one expects generalized solutions to piecewise-smooth systems to be continuous functions of time, because these systems do not allow for discontinuous jumps in their (continuous) state, but with the differentiability requirement for the solution (in some way) appropriately relaxed. Solutions for more general classes of hybrid systems (which may allow discontinuous jumps in the state) are trickier, and require generalized time domains, such as hybrid time domains explored in the work of Sanfelice, Goebel and Teel [36, 18]. In our approach, we will not directly make use of these notions, relying instead on the semantics of hybrid automata (after Lygeros et al. [27]), which we shall describe presently.

2.3 Hybrid automata as operational models

Hybrid automata were first introduced by Alur et al. [1] as a formal specification language for hybrid systems. They provide operational models for hybrid systems in the same way that transition systems provide models for discrete computer programs, making it possible to give a precise mathematical description of their *execution.* We will employ the term *evolution* when speaking about hybrid *systems* (just as with continuous systems) and use the term *execution* only in the context of operational models, such as hybrid automata.

As formal models, hybrid automata have been used extensively in both modelling [12] and verification of properties

[2] By this we understand a *finite* expression in terms of polynomials and elementary special functions such as sin, cos, exp, ln, etc.

[3] E.g. it is continuity of the right-hand side that guarantees the existence of solutions (by Peano's theorem).

in hybrid systems [14, 40]. Below we reproduce a very convenient definition of hybrid automata and their execution, due to Lygeros et al. [27]; for alternative definitions the interested reader is invited to consult [1, 22, 42].

Definition 1. *Formally, a* hybrid automaton HA *is given by an* 8*-tuple*

$$\mathrm{HA} = (Q, X, F, \mathrm{Init}, \mathrm{Dom}, E, G, R),$$

where the elements are as follows:

- $Q = \{q_1, q_2, \ldots, q_m\}$ *is a finite set of* discrete states*,*
- $X = \mathbb{R}^n$ *is a set of* continuous states*,*
- $F : Q \times X \to \mathbb{R}^n$ *is a* vector field*,*
- $\mathrm{Init} \subseteq Q \times X$ *is a set of* initial states*,*
- $\mathrm{Dom} : Q \to 2^X$ *is a* mode domain *(also* invariant*),*
- $E \subseteq Q \times Q$ *is a set of* edges *(also* discrete transitions*),*
- $G : E \to 2^X$ *is a* guard condition*,*
- $R : E \times X \to 2^X$ *is a* reset map*.*

Standard assumptions with this definition are that guard conditions are non-empty whenever they are specified, i.e. for all $e \in E$ it is the case that $G(e) \neq \emptyset$ and also that reset maps can only take the system to a genuine continuous state, i.e. for all $x \in G(e)$, $R(e, x) \neq \emptyset$.

2.3.1 Semantics of hybrid automata

A *hybrid time trajectory* is a finite or infinite sequence of contiguous time intervals starting at 0, where the end points are interpreted as times at which a discrete *event*, such as a transition, occurs. More formally, following [27], a hybrid time trajectory is a sequence of intervals $\tau = \{I_i\}_{i=0}^N$, for which $I_i = [\tau_i, \tau'_i]$ for all $i < N$, where $N \in \mathbb{N} \cup \{\infty\}$, and $\tau_i \leq \tau'_i = \tau_{i+1}$ for all i. If the sequence is finite, i.e. if $N < \infty$, then either $I_N = [\tau_N, \tau'_N]$ or $I_N = [\tau_N, \tau'_N)$. Intuitively, one may think of τ_i as the times at which discrete transitions occur.

An *execution* (or a *run*) of a hybrid automaton is defined to be the triple $(\tau, q, \varphi^i_t(\boldsymbol{x}))$, where τ is a hybrid time trajectory, $q : \langle\tau\rangle \to Q$, where $\langle\tau\rangle$ is defined to be the set $\{0, 1, \ldots, N\}$ if τ is finite and $\{0, 1, \ldots\}$ otherwise [27], is a map and $\varphi^i_t(\boldsymbol{x})$ is a collection of differentiable functions $\varphi^i_t(\boldsymbol{x}) : I_i \to \mathbb{R}^n$ such that $(q(0), \varphi^0_0(\boldsymbol{x})) \in \mathrm{Init}$ and for all $t \in [\tau_i, \tau'_i)$ it is the case that $\dot{\boldsymbol{x}} = F(q(i), \varphi^i_t(\boldsymbol{x}))$ and $\varphi^i_t(\boldsymbol{x}) \in \mathrm{Dom}(q(i))$. It is also required that transitions respect the guards and the reset maps, i.e. $e = (q(i), q(i+1)) \in E$, $\varphi^i_{\tau'_i}(\boldsymbol{x}) \in G(e)$ and $(\varphi^i_{\tau'_i}(\boldsymbol{x}), \varphi^{i+1}_{\tau_{i+1}}(\boldsymbol{x})) \in R(e)$.

3. PROBLEM OVERVIEW

This section gives an overview of the challenges associated with modelling piecewise-smooth systems using the hybrid automaton formalism.

If one were to naïvely translate a system of the form shown in (2) into a hybrid automaton, as a first step one could simply take the sets $M_1, \ldots, M_m$ to be the mode invariants of the discrete states in the automaton (i.e. by letting Dom in Definition 1 be $q_i \mapsto M_i$ for each $i = 1, \ldots, m$) and set the continuous dynamics within these modes to be governed by the differential equation $\dot{\boldsymbol{x}} = \mathfrak{f}_i(\boldsymbol{x})$, i.e. letting the vector field $F(q_i, \boldsymbol{x}) = \mathfrak{f}_i(\boldsymbol{x})$ for each $i = 1, \ldots, m$, respectively. The resulting hybrid automaton would have $|Q| = m$ discrete states and no discrete transitions between them. Clearly, this would not be an adequate model, since the original system will most likely evolve into and out of the sets $M_1, \ldots, M_m$. This fact raises an immediate problem: in order to have discrete transitions in the hybrid automaton one is required to specify their enabling guards, i.e. sets of states *within the mode invariant of the outgoing discrete state* in which a discrete transition is possible.

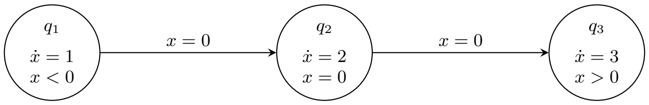


Figure 1: Naïve construction (mode transitions impossible).

To appreciate the problem more fully, let us consider a simple 1-dimensional system defined on the partition of the real line $\mathbb{R}$ into three regions: $x < 0$, $x = 0$ and $x > 0$, and where the vector fields are respectively given by $\mathfrak{f}_1(x) = 1$, $\mathfrak{f}_2(x) = 2$ and $\mathfrak{f}_3(x) = 3$ (i.e. $\dot{x} = 1$, $\dot{x} = 2$, and $\dot{x} = 3$) inside each region. Clearly, one expects this system, when started inside $x < 0$, to transition into $x = 0$ and then to $x > 0$. In order for a hybrid automaton to faithfully model the behaviour of this system, we require two discrete transitions that take the state from $x < 0$ to $x = 0$ and from $x = 0$ to $x > 0$; however, in the former transition it is not possible to specify $x = 0$ to be the guard (as shown in Fig. 1), since this set lies outside of the mode invariant $x < 0$ of the outgoing discrete state. It is possible to declare the transition guard to be in some thin layer near the boundary, e.g. $\delta < x < 0$, where $-\frac{1}{\delta}$ is large, but any such choice of δ would be rather arbitrary in the general case. Furthermore, there would remain another important problem, this time with the latter transition from $x = 0$ to $x > 0$: in order to make such a transition without creating discontinuities (in this case "gaps") in the trajectory through reset maps, the state of the system needs to lie within the mode invariant of the destination discrete state when the transition guard is enabled. The guard $x = 0$ is thus also unsuitable in this case and there is no easy fix to this problem.

Instead of using $M_1, \ldots, M_m$ as mode invariants in the automaton, one may instead opt to use their closures $\overline{M_1}, \ldots, \overline{M_m}$ with a view to enabling the transition guards on appropriate subsets of the boundaries $\partial M_1, \ldots, \partial M_m$, which would now lie inside the corresponding mode invariants. This approach, while conceptually simple, has a number of serious deficiencies and results in hybrid automata that exhibit *chattering runs*, i.e. can perform an arbitrary number of discrete transitions without advancing the continuous state or time.

The use of set closures additionally overlooks an important computational drawback, which is that closures are typically very difficult to compute exactly for important classes for sets, such as e.g. *semi-algebraic sets* (i.e. sets described by a finite Boolean combination of polynomial equalities and inequalities; see e.g. [28, Definition 8.6.1]).

Remark 1. *In general for semi-algebraic sets,* $\overline{S}$ *cannot be obtained from* S *by syntactically replacing every instance of strict inequalities in its description by non-strict inequalities (e.g.* $x^3 - x^2 \geq 0$ *is* ***not*** *the closure of* $x^3 - x^2 > 0$*) [3, Remark 3.2]. The closure of a semi-algebraic set* S *is given* *by the set*

$$\overline{S} = \{\boldsymbol{x} \in \mathbb{R}^n \mid \forall\, r > 0.\ \exists\, \boldsymbol{y} \in S.\ \|\boldsymbol{y} - \boldsymbol{x}\|^2 < r^2\},$$

where the norm $\|\|$ is the standard Euclidean distance (see e.g. [3, Chapter 3]). Let the set S be described by a quantifier-free formula in the theory of real arithmetic with free variables $x_1, \ldots, x_n$. By performing a syntactic substitution of the free variables x_i by y_i ($i = 1, \ldots, n$) everywhere in the formula, one obtains a quantifier-free formula in the variables $y_1, \ldots, y_n$. The closure $\overline{S}$ can then be characterized by the formula

$$\forall\, r > 0.\ \exists\, y_1, \ldots, y_n.\ S \wedge (y_1 - x_1)^2 + \cdots + (y_n - x_n)^2 < r^2,$$

where $x_1, \ldots, x_n$ are treated as fresh free variables and r is a fresh bound variable. It is therefore possible to apply real quantifier elimination to reduce this formula to an equivalent one that is quantifier-free and features only the free variables $x_1, \ldots, x_n$.

Real quantifier elimination (QE) is computationally expensive, having complexity doubly-exponential in the number of quantifier alternations [10]. The popular CAD *algorithm for real QE, is doubly-exponential in the number of variables [4], which makes it impractical for problems with a large number of variables (using currently existing implementations).*

In this paper we pursue a very different approach to constructing HA operational models of PWS, which does not require computing set closures , but instead requires only the "relevant" subsets on their boundaries and relies fundamentally on the notion of "entry" and "exit" sets that will be the subject of the following section.

4. OPERATIONAL MODELS

This section will review some important definitions before presenting an algorithm for automatically generating HA operational models of PWS.

4.1 Fundamental Definitions

We start by defining an important set that will shortly become of interest:

Definition 2 (Inward Crossing Set).

$$\mathrm{Enter}_{\mathfrak{f}}(S) \equiv \{\boldsymbol{x} \in \mathbb{R}^n \mid \exists\, \varepsilon > 0.\ \forall\, t \in (0, \varepsilon).\ \varphi_t(\boldsymbol{x}) \in S\ \wedge\ \forall\, t \in (-\varepsilon, 0).\ \varphi_t(\boldsymbol{x}) \notin S\}$$

where $\varphi_t(\boldsymbol{x})$ denotes the (unique) solution to the locally Lipschitz-continuous system of ODEs $\dot{\boldsymbol{x}} = \mathfrak{f}(\boldsymbol{x})$.

The intuition, as suggested by the name, is that $\mathrm{Enter}_{\mathfrak{f}}(S)$ describes the states at which the system is about to evolve inside S, after having only just evolved outside of S. Likewise, we define $\mathrm{Exit}_{\mathfrak{f}}(S)$ to be the set of states at which the system is about to evolve outside of S, after having only just evolved inside, i.e. $\mathrm{Exit}_{\mathfrak{f}}(S) \equiv \mathrm{Enter}_{\mathfrak{f}}(\neg S)$, where $\neg S := \mathbb{R}^n \setminus S$. Note that such states need not necessarily lie within S itself and may lie outside; however, they necessarily lie on the boundary of S. We observe that the crossing set, by its very definition, can be expressed by means of one fundamental building block.

Lemma 1 (Crossing Set Deconstruction). $\mathrm{Enter}_{\mathfrak{f}}(S) \equiv \mathrm{In}_{\mathfrak{f}}(S)\ \cap\ \mathrm{In}_{-\mathfrak{f}}(\neg S)$, *where*

$$\mathrm{In}_{\mathfrak{f}}(S) \equiv \{\boldsymbol{x} \in \mathbb{R}^n \mid \exists\, \varepsilon > 0.\ \forall\, t \in (0, \varepsilon).\ \varphi_t(\boldsymbol{x}) \in S\}.$$

Intuitively, $\mathrm{In}_{\mathfrak{f}}(S)$ denotes the states in $\mathbb{R}^n$ from which the motion of the system takes place within the set S for some time segment immediately following 0 (i.e. in the immediate future). By analogy, when considering $-\mathfrak{f}$, the reverse of the vector field $\mathfrak{f}$, $\mathrm{In}_{-\mathfrak{f}}(S)$ denotes the states in $\mathbb{R}^n$ from which the motion of the system took place within the set S for some time segment immediately preceding 0 (i.e. in the immediate past).

In the special case when the system $\dot{\boldsymbol{x}} = \mathfrak{f}(\boldsymbol{x})$ has polynomial right-hand sides and S is a semi-algebraic set, the sets $\mathrm{In}_{\mathfrak{f}}(S)$, and hence $\mathrm{In}_{-\mathfrak{f}}(S)$, are also semi-algebraic and can be computed exactly (a result due to Liu et al. [25]). As a consequence, the sets $\mathrm{Enter}_{\mathfrak{f}}(S)$ and $\mathrm{Exit}_{\mathfrak{f}}(S)$ are also computable and semi-algebraic under these assumptions.

We stress the fact that the boundary of S need not be included in $\mathrm{Enter}_{\mathfrak{f}}(S) \cup \mathrm{Exit}_{\mathfrak{f}}(S)$. In particular, the set

$$\mathrm{Bounce}_{\mathfrak{f}}(S) \equiv \mathrm{In}_{\mathfrak{f}}(S)\ \cap\ \mathrm{In}_{-\mathfrak{f}}(S)$$

describes those states that may leave S momentarily at a point while evolving within S before and after the "bounce" and can therefore lie outside of $\mathrm{Enter}_{\mathfrak{f}}(S) \cup \mathrm{Exit}_{\mathfrak{f}}(S)$. Appendix B provides an illustration to help develop some intuition about the meaning of these sets.

4.2 Generating Hybrid Automata

We now have at our disposal the machinery necessary for building operational models of piecewise-smooth systems $\dot{\boldsymbol{x}} = \mathfrak{F}(\boldsymbol{x})$, i.e. systems of the form:

$$\dot{\boldsymbol{x}} = \begin{cases} \mathfrak{f}_1(\boldsymbol{x}) & \boldsymbol{x} \in M_1\,, \\ \quad\vdots & \\ \mathfrak{f}_m(\boldsymbol{x}) & \boldsymbol{x} \in M_m\,. \end{cases}$$

Given such a system, our aim is to synthesize a hybrid automaton that provides an adequate model of the behaviour of the system. To do this, our approach we will be to first *augment* the original invariant modes of the system M_i with additional states, before they can become mode invariants of a hybrid automaton. This step requires a definition.

Definition 3. *Given a semi-algebraic set $S \subseteq \mathbb{R}^n$ and a system of polynomial ODEs $\dot{\boldsymbol{x}} = \mathfrak{f}(\boldsymbol{x})$, the* augmented set *of S with respect to this system,* $\mathrm{Aug}(S, \mathfrak{f})$*, is defined by*

$$\mathrm{Aug}(S, \mathfrak{f}) \equiv S \cup \mathrm{Enter}_{\mathfrak{f}}(S) \cup \mathrm{Exit}_{\mathfrak{f}}(S) \cup \mathrm{Bounce}_{\mathfrak{f}}(S)\,.$$

In the context of piecewise-smooth systems of the form $\dot{\boldsymbol{x}} = \mathfrak{F}(\boldsymbol{x})$, whenever we wish to augment the set M_i with respect to the system $\dot{\boldsymbol{x}} = \mathfrak{f}_i(\boldsymbol{x})$, we shall adopt a more concise notation and simply write $\mathfrak{M}_i$, i.e. $\mathfrak{M}_i \equiv \mathrm{Aug}(M_i, \mathfrak{f}_i)$.

The definition extends each invariant mode S with its "entry", "exit" and "bounce" sets. The main intuition being that if the system was to enter or exit the mode invariant S with respect to the dynamics $\mathfrak{f}$ then it will do so by necessarily crossing those sets. The set $\mathrm{Bounce}_{\mathfrak{f}}(S)$ allows the evolution to continue within S after "momentarily exiting" S. If in addition the mode invariants have to satisfy a global constraint M, then it should be accounted for by intersecting it with $\mathrm{Aug}(M_i, \mathfrak{f}_i)$.

Algorithm 1 gives a pseudocode procedure for generating a hybrid automaton $\mathrm{HA}_{\mathfrak{F}}$. The procedure begins constructing the automaton by first creating m distinct discrete states $Q = \{q_1, \ldots, q_m\}$ (line 1), defining X to be $\mathbb{R}^n$ (line 2) and creating a set of edges (i.e. transitions) E by computing the Cartesian product $Q \times Q$ and removing all edges of the form (q_i, q_i), i.e. removing all stuttering/self-looping transitions (line 3). It then proceeds to initially assign the empty set to all the remaining variables on line 4. The algorithm then proceeds to create the modes of the hybrid automaton $\mathrm{HA}_{\mathfrak{F}}$ in its first loop (lines 5–12), where it builds the extensional definition of F by assigning the vector field $\mathfrak{f}_i$ to the discrete state q_i (line 6), augmenting each set M_i with its "entry", "exit" and "bounce" sets (line 7) and using this to build an extensional definition of the Dom mapping (line 8) which provides the mode invariant $\mathfrak{M}_i$ for each state q_i of the hybrid automaton. Lines 9–11 are responsible for converting the initial set of states for the PWS into one for the hybrid automaton (this step can in fact be factored out of the algorithm and performed separately).

The second loop of the algorithm (lines 13–16) constructs the discrete transitions and is responsible for defining the discrete transition behaviour of the resulting automaton. The loop iterates through all the transitions constructed on line 3 and defines the guards (line 14) and reset maps (line 15) associated with each transition. The reset map is chosen to be the identity and therefore does not affect the state of the system upon taking any transition. Different possible choices for the guard condition $\mathrm{GC}(i, j)$ (line 14) are discussed in the next section.

```
Data: $M \subseteq \mathbb{R}^n, M_1, \ldots, M_m \subseteq M, \mathfrak{f}_1, \ldots, \mathfrak{f}_m : \mathbb{R}^n \to \mathbb{R}^n, X_0 \subseteq M$
Result: Hybrid automaton $\mathrm{HA}_{\mathfrak{F}}$
1  $Q \leftarrow \{q_1, \ldots, q_m\}$;
2  $X \leftarrow \mathbb{R}^n$;
3  $E \leftarrow Q \times Q \setminus \{(q_1, q_1), (q_2, q_2), \ldots (q_m, q_m)\}$;
4  $\mathrm{Init}, \mathrm{Dom}, F, G, R \leftarrow \emptyset$;
5  foreach $i \in \{1, \ldots, m\}$ do
6      $F \leftarrow F \cup \{((q_i, \boldsymbol{x}), \mathfrak{f}_i(\boldsymbol{x}))\}$;
7      $\mathfrak{M}_i \leftarrow \mathrm{Aug}(M_i, \mathfrak{f}_i) \cap M$;
8      $\mathrm{Dom} \leftarrow \mathrm{Dom} \cup \{q_i \mapsto \mathfrak{M}_i\}$;
9      if $X_0 \cap M_i \neq \emptyset$ then
10         $\mathrm{Init} \leftarrow \mathrm{Init} \cup \{(q_i, \boldsymbol{x}) \mid \boldsymbol{x} \in M_i \cap X_0\}$
11     end
12 end
13 foreach $e = (q_i, q_j) \in E$ do
14     $G \leftarrow G \cup \{(e, \mathrm{GC}(i, j))\}$;
15     $R \leftarrow R \cup \{((q_i, q_j), \boldsymbol{x} \mapsto \{\boldsymbol{x}\})\}$
16 end
17 return $(Q, X, F, \mathrm{Init}, \mathrm{Dom}, E, G, R)$
```

Algorithm 1: Procedure for synthesizing a HA from PWS.

4.3 Discrete Transition Behaviour

The transition guard $G(e) = \mathrm{GC}(i, j)$, $i \neq j$, for the transition $q_i \to q_j$ entirely determines the discrete transition behaviour of the automaton. In what follows, we will consider three choices for this formula.

REMARK 2. *We stress the fact that these are by no means the only possible semantics; they are primarily meant to exemplify how the method works and how one can adapt Algorithm 1 to generate operational models exhibiting qualitatively different behaviours.*

Recall that mode q_i (resp. q_j) has mode invariant $\mathfrak{M}_i$ (resp. $\mathfrak{M}_j$).

$$\begin{aligned} \mathrm{I} &\equiv \ \mathfrak{M}_i \wedge \mathfrak{M}_j \wedge \mathrm{In}_{\mathfrak{f}_j}(M_j) \\ \mathrm{II} &\equiv \ \mathrm{I} \wedge \neg \mathrm{Enter}_{\mathfrak{f}_i}(M_i) \wedge \neg \mathrm{Bounce}_{\mathfrak{f}_j}(M_j) \\ \mathrm{III} &\equiv \ \mathrm{I} \wedge \neg \mathrm{In}_{\mathfrak{f}_i}(M_i) \end{aligned}$$

Informally, these formulas are characterizing the sets of states where (1) the augmented mode invariants $\mathfrak{M}_i$ and $\mathfrak{M}_j$ intersect to allow for continuous transitions and (2) where the trajectory of the system in mode q_j can evolve within that mode for some time, hence the intersection with $\mathrm{In}_{\mathfrak{f}_j}(M_j)$. Formulas II and III impose additional constraints on the guard. Namely, formula II additionally requires that the guard does not feature states in the intersection of the "entering" set of the outgoing state and the "bounce" set of the incoming state. As will be seen in later sections, this is primarily done to eliminate so-called *chattering* in the model. Formula III is different in that it only enables a transition guard if no further continuous motion is possible within the mode. This has the effect that transitions *must* be taken precisely when they are enabled.

Replacing $\mathrm{GC}(i, j)$ in line 14 of Algorithm 1 by formula I, II or III will generally result in a different operational model which can exhibit very different behaviour. In what follows, we will refer to these formulas as respectively defining guard conditions of *type* I, II and III.

4.4 Computability

An operational model of a PWS in the form of a hybrid automaton is computable using Algorithm 1 whenever the vector fields $\mathfrak{f}_1, \ldots, \mathfrak{f}_m$ are polynomial and the sets $M_1, \ldots, M_m, M$ and X_0 are semi-algebraic.

We recall that a set is semi-algebraic if it is characterized by a finite Boolean combination of polynomial equations and inequalities. Thus, the formula $x_1 > 0 \wedge x_2 = 0 \vee x_2^3 - x_1 \leq 0$, where the symbols x_1, x_2 are interpreted over the real numbers, characterizes the semi-algebraic set $\{(x_1, x_2) \in \mathbb{R}^2 \mid x_1 > 0 \wedge x_2 = 0 \vee x_2^3 - x_1 \leq 0\}$. It suffices to consider formulas without quantifiers, e.g. $\forall$ and $\exists$, since the theory of real arithmetic admits quantifier elimination [38] and therefore any formula featuring quantifiers may be reduced to an equivalent quantifier-free formula using a terminating procedure.[4]

It was shown in [25] that the set $\mathrm{In}_{\mathfrak{f}}(S)$ can be computed exactly by employing higher-order *Lie derivatives* and the *ascending chain property* of Noetherian rings. A Lie derivative of a polynomial $p : \mathbb{R}^n \to \mathbb{R}$ with respect to the polynomial vector field $\mathfrak{f} : \mathbb{R}^n \to \mathbb{R}^n$ is also a polynomial denoted $\mathfrak{L}_{\mathfrak{f}}(p)$ and defined as $\mathfrak{L}_{\mathfrak{f}}(p) \equiv \nabla p \cdot \mathfrak{f} = \sum_{i=1}^n \frac{\partial p}{\partial x_i} f_i$. It gives the total derivative of the p with respect to time, i.e. the rate of change of p along the solutions to the corresponding system of ODEs. Higher-order Lie derivatives are defined inductively as $\mathfrak{L}_{\mathfrak{f}}^k(p) = \mathfrak{L}_{\mathfrak{f}}(\mathfrak{L}_{\mathfrak{f}}^{k-1}(p))$, with $\mathfrak{L}_{\mathfrak{f}}^0(p) = p$.

In addition to [25], a description of the main idea behind the procedure for constructing $\mathrm{In}_{\mathfrak{f}}(S)$ may be found in [17, Section 5.4]; a brief sketch of this construction is also given in Appendix A of this article. Similar ideas employing higher-order Lie derivatives and ascending chains of ideals have also appeared elsewhere, e.g. [33, 16]. As a consequence,

[4] A number of algorithms have been developed since Tarski's [38] and Seidenberg's [37] seminal papers, e.g. the CAD algorithm due to Collins [6].

the sets $\mathrm{Enter}_{\mathfrak{f}}(S)$, $\mathrm{Exit}_{\mathfrak{f}}(S)$ and $\mathrm{Bounce}_{\mathfrak{f}}(S)$ are also semi-algebraic and may be computed exactly using a terminating algorithm.

5. DYNAMIC PROPERTIES OF OPERATIONAL MODELS

This section will illustrate some of the dynamic phenomena observed in the operational models that we can compute using Algorithm 1 and will discuss some of the differences in their behaviour when different types of guard conditions are employed.

5.1 Non-determinism

Non-determinism occurs when the piecewise-smooth system may evolve inside more than one of its modes. At first sight, this may look surprising because in a PWS any state $\boldsymbol{x} \in M$ belongs to exactly one region M_i, $i \in \{1, \dots, m\}$, if one indeed has a mathematical partition of M into these regions, and therefore there cannot be any ambiguity in the choice of the ODEs that should govern the continuous state evolution at $\boldsymbol{x}$. However, generalized solutions to the system at $\boldsymbol{x}$ may not be unique even when the ODEs inside each mode all have unique solutions when considered separately. This is mirrored in our operational models, where we augment the regions M_i with their respective "entry" and "exit" sets to obtain the augmented mode invariants $\mathfrak{M}_i$ in the hybrid automaton. One may face a scenario where $\boldsymbol{x} \in \mathfrak{M}_i$ and $\boldsymbol{x} \in \mathfrak{M}_j$, with $i \neq j$, and both transition guards between the two states q_i and q_j are enabled. For instance, $\boldsymbol{x}$ may lie in a region where both $\mathfrak{M}_i \wedge \mathfrak{M}_j \wedge \mathrm{In}_{\mathfrak{f}_i}(M_i)$ and $\mathfrak{M}_i \wedge \mathfrak{M}_j \wedge \mathrm{In}_{\mathfrak{f}_j}(M_j)$ hold true.

The standard semantics of transition guards of hybrid automata is that they *enable* transitions, but do not force them (this is known as *non-urgent*, or *may* semantics [13]). Thus, while there is no ambiguity about the initial discrete state of the hybrid automaton for any given $\boldsymbol{x} \in M$, the system is free to take an enabled transition immediately after it starts evolving. This non-determinism can be informally understood as capturing the "instability" that arbitrarily small perturbations in the initial state can cause in the mode switching behaviour of the piecewise-smooth system.

5.2 Chattering Runs

A phenomenon known as *chatter* is traditionally associated with so-called *Zeno behaviour* that can occur in mathematical models of hybrid systems and can present a problem for their simulation and verification. This behaviour is non-physical and manifests itself in the possibility of performing an infinite number of transition in a finite amount of time.

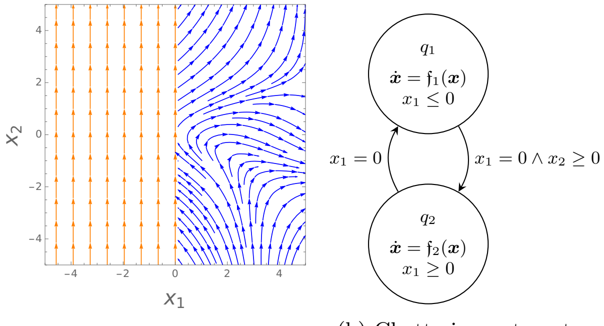
For example, a hybrid automaton will admit *chattering runs* whenever for two distinct states q_i and q_j there are transitions in both directions such that their respective transition guards have non-empty intersection. Any state $\boldsymbol{x}$ within this intersection can shuttle back and forth between the states q_i and q_j an arbitrary (though perhaps not infinite) number of times.

As an example, let us consider a PWS with two modes:

$$\dot{\boldsymbol{x}} = \mathfrak{f}_1(\boldsymbol{x}) \equiv \begin{cases} \dot{x}_1 = 0\,, \\ \dot{x}_2 = x_2^2 + 2\,, \end{cases} \qquad x_1 \leq 0\,,$$

$$\dot{\boldsymbol{x}} = \mathfrak{f}_2(\boldsymbol{x}) \equiv \begin{cases} \dot{x}_1 = x_1 + 4x_2 - x_1x_2\,, \\ \dot{x}_2 = x_2^2 - x_1 + 2\,, \end{cases} \qquad x_1 > 0.$$

By running Algorithm 1 with guard conditions of type I, one obtains a hybrid automaton shown in Fig. 2b. This automaton admits chattering runs because on the set characterized by $x_1 = 0 \wedge x_2 \geq 0$ the guards for transitions between both modes are enabled simultaneously and the system may thus shuttle back and forth arbitrarily may times without advancing in (continuous) time. However, if one were to employ guard conditions of type II, the resulting automaton (Fig. 3) would be chatter-free.



(a) Phase portrait.

(b) Chattering automaton.

Figure 2: Chattering in the presence of non-determinism.

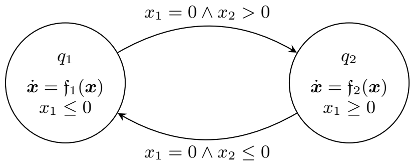

$x_1 = 0 \wedge x_2 > 0$
q_1
$\dot{\boldsymbol{x}} = \mathfrak{f}_1(\boldsymbol{x})$
$x_1 \leq 0$
q_2
$\dot{\boldsymbol{x}} = \mathfrak{f}_2(\boldsymbol{x})$
$x_1 \geq 0$
$x_1 = 0 \wedge x_2 \leq 0$

Figure 3: Chatter-free automaton.

Since infinite Zeno executions cannot in practice be realized, it is common to consider only the *non-Zeno executions* when modelling systems using hybrid automata [22, 11] (this is also the case with hybrid programs [35]).

We should note that infinite chattering runs are a special kind of Zeno behaviour, which some authors distinguish from the more involved *genuine Zeno* behaviour (see e.g. [2]). Chatter-free automata may still suffer from this latter type of Zeno behaviour. Detecting and eliminating genuine Zeno behaviour in hybrid automata is highly non-trivial and is the focus of ongoing research.

5.3 Mythical Modes

A piecewise-smooth system may feature a mode M_i inside which it is altogether impossible to evolve continuously according to its respective system of ODEs $\dot{\boldsymbol{x}} = \mathfrak{f}_i(\boldsymbol{x})$. More precisely, it is possible that $M_i \cap \mathrm{In}_{\mathfrak{f}_i}(M_i) = \emptyset$. Inside such a mode, the (continuous) state of the system remains invariant and may only change by switching into a different mode; such a mode is sometimes called *mythical* [30, 31]. For example, in a system where the state space is the real line $\mathbb{R}$

that is partitioned into 3 modes $x < 0$, $x = 0$ and $x > 0$ where the dynamics is respectively $\dot{x} = 1$, $\dot{x} = 2$ and $\dot{x} = 3$, the mode $x = 0$ is mythical.

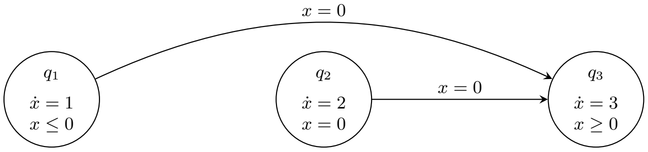


Figure 4: Mythical mode q_2.

Following our approach, the mode invariants for the hybrid automaton are augmented to be $x \leq 0$, $x = 0$ and $x \geq 0$ respectively, and a transition from $x \leq 0$ into $x \geq 0$ is possible without ever visiting the mythical mode. In general, in hybrid automata constructed using our method (e.g. Fig. 4 where only possible transitions are depicted with their guards) it is impossible to transition into mythical modes with any of the three types of guard conditions.

5.4 Sliding Modes

In control systems literature, it is not uncommon to encounter systems of the form

$$\dot{\boldsymbol{x}} = \begin{cases} \mathfrak{f}_1(\boldsymbol{x}) & s(\boldsymbol{x}) > 0\,, \\ \mathfrak{f}_2(\boldsymbol{x}) & s(\boldsymbol{x}) < 0\,, \end{cases}$$

where $s : \mathbb{R}^n \to \mathbb{R}$ is some differentiable (often polynomial) function. These and similar systems are sometimes termed *variable structure systems* (VSS) and have been applied in discontinuous non-linear control strategies, known as *variable structure control* (VSC). A phenomenon known as *sliding motion* lies at the heart of an important class of variable structure control, known as *sliding mode control* (SMC), which, broadly speaking, achieves system order reduction by steering the trajectories of an n-dimensional system onto an $n - 1$ dimensional switching hyper-surface in the system's state space, defined by $s = 0$. The so-called *sliding motion* taking place on the hyper-surface corresponds to the infinitely-fast switching between the modes governing the evolution on either side of the surface [45], i.e. inside regions where $s > 0$ and $s < 0$.

Remark 3. *Note however, that the description of the system may not explicitly prescribe any dynamics on the* switching surface $s = 0$ *itself.*

In practice, sliding motions are often modelled by introducing a so-called *equivalent control* [41] on the switching surface; this is achieved by letting

$$\dot{\boldsymbol{x}} = \mathfrak{f}_s(\boldsymbol{x}) = \frac{\mathfrak{f}_1(\boldsymbol{x}) + \mathfrak{f}_2(\boldsymbol{x})}{2} + u_{eq}\frac{\mathfrak{f}_1(\boldsymbol{x}) - \mathfrak{f}_2(\boldsymbol{x})}{2}\,,$$

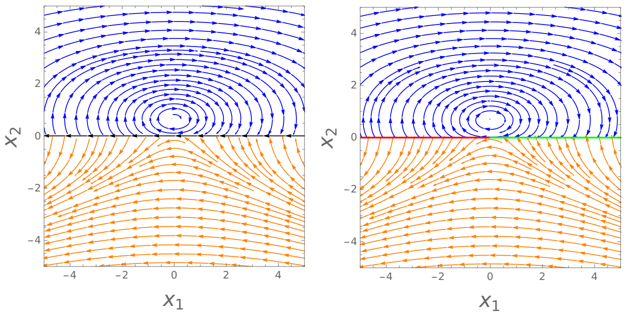
where $u_{eq} = \dfrac{\mathfrak{L}_{\mathfrak{f}_1}(s) \;+\; \mathfrak{L}_{\mathfrak{f}_2}(s)}{\mathfrak{L}_{\mathfrak{f}_2}(s) \;-\; \mathfrak{L}_{\mathfrak{f}_1}(s)}$, be the sliding dynamics on the surface $s = 0$ (e.g. see [32]).

Let us consider a 2-dimensional non-linear system with a 1-dimensional sliding mode that was obtained by applying an equivalent control. The system is given by:

$$\dot{\boldsymbol{x}} = \mathfrak{f}_1(\boldsymbol{x}) \equiv \begin{cases} \dot{x}_1 = x_2^3 + \frac{3x_2^2}{8} + \frac{3x_2}{64} - \frac{255}{512}\,, \\ \dot{x}_2 = -\frac{x_1}{8} - x_1 x_2\,, \end{cases} \qquad x_2 > 0\,,$$

$$\dot{\boldsymbol{x}} = \mathfrak{f}_2(\boldsymbol{x}) \equiv \begin{cases} \dot{x}_1 = -2x_2^3 + \frac{9x_2^2}{8} + \frac{123x_2}{320} - \frac{303}{640}\,, \\ \dot{x}_2 = 0\,, \end{cases} \qquad x_2 = 0\,,$$

$$\dot{\boldsymbol{x}} = \mathfrak{f}_3(\boldsymbol{x}) \equiv \begin{cases} \dot{x}_1 = x_2^3 - \frac{3x_2^2}{2} + \frac{3x_2}{4} - \frac{3}{8}\,, \\ \dot{x}_2 = \frac{x_1}{2} - x_1 x_2\,, \end{cases} \qquad x_2 < 0\,.$$



(a) Phase portrait.

(b) Stable and unstable sliding.

Figure 5: Piecewise-smooth system $\dot{\boldsymbol{x}} = \mathfrak{F}(\boldsymbol{x})$ with a sliding mode at $x_2 = 0$ that is unstable when $x_1 > 0$ (shown in red) and a stable when $x_1 < 0$ (in green).

Sliding occurs on the set characterized by $x_2 = 0$ and $\dot{\boldsymbol{x}} = \mathfrak{f}_2(\boldsymbol{x})$ is the equivalent control dynamics which steers the system along the surface $x_2 = 0$ (Fig. 5a). The system exhibits both stable and unstable sliding behaviour, which can be observed in the phase portrait, as shown in Fig. 5b. Roughly speaking, in the neighbourhoods of states where the sliding mode is stable the vector fields are "pointing towards" the sliding set, whereas in the neighbourhood of states where it is unstable the vector fields are "pointing outwards" away from the set.

For this system, different types of guard conditions lead to radically different operational models. The resulting hybrid automata employing guard conditions of type I, II and III are respectively shown in Fig. 6, Fig. 7 and Fig. 8.

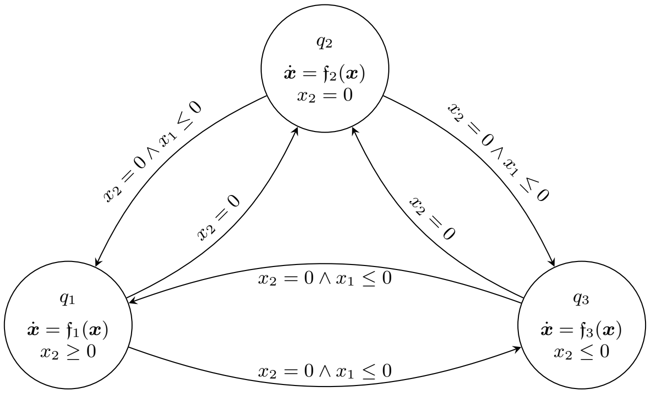

q_2
$\dot{\boldsymbol{x}} = \mathfrak{f}_2(\boldsymbol{x})$
$x_2 = 0$
$x_2 = 0 \wedge x_1 \leq 0$
$x_2 = 0$
$x_2 = 0$
$x_2 = 0 \wedge x_1 \leq 0$
q_1
$\dot{\boldsymbol{x}} = \mathfrak{f}_1(\boldsymbol{x})$
$x_2 \geq 0$
$x_2 = 0 \wedge x_1 \leq 0$
$x_2 = 0 \wedge x_1 \leq 0$
q_3
$\dot{\boldsymbol{x}} = \mathfrak{f}_3(\boldsymbol{x})$
$x_2 \leq 0$

Figure 6: Hybrid automaton model with guard conditions of type I.

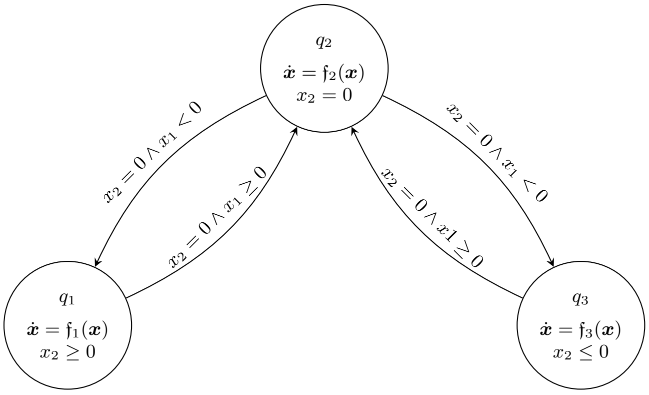

q_2
$\dot{\boldsymbol{x}} = \mathfrak{f}_2(\boldsymbol{x})$
$x_2 = 0$
$x_2 = 0 \wedge x_1 < 0$
$x_2 = 0 \wedge x_1 \geq 0$
$x_2 = 0 \wedge x_1 \geq 0$
$x_2 = 0 \wedge x_1 < 0$
q_1
$\dot{\boldsymbol{x}} = \mathfrak{f}_1(\boldsymbol{x})$
$x_2 \geq 0$
q_3
$\dot{\boldsymbol{x}} = \mathfrak{f}_3(\boldsymbol{x})$
$x_2 \leq 0$

Figure 7: Hybrid automaton model with guard conditions of type II.

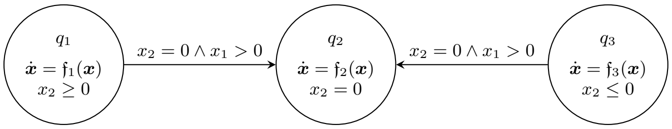

q_1
$\dot{\boldsymbol{x}} = \mathfrak{f}_1(\boldsymbol{x})$
$x_2 \geq 0$
$x_2 = 0 \wedge x_1 > 0$
q_2
$\dot{\boldsymbol{x}} = \mathfrak{f}_2(\boldsymbol{x})$
$x_2 = 0$
$x_2 = 0 \wedge x_1 > 0$
q_3
$\dot{\boldsymbol{x}} = \mathfrak{f}_3(\boldsymbol{x})$
$x_2 \leq 0$

Figure 8: Hybrid automaton model with guard conditions of type III.

The three automata differ in the way they model non-determinism in the system. In particular guard conditions of type III result in the automaton in Fig. 8, which is completely deterministic and only models the stable sliding taking place in the system; there is no non-determinism corresponding to unstable sliding in this operational model. In practice, this behaviour is un-physical because unstable motions can leave the unstable sliding mode under arbitrarily small perturbations in the state or the vector field. As such, this operational model represents a mathematical idealization which is of little use when modelling physical systems. However, if physical considerations are unimportant, the model is interesting because it has the property that discrete transitions are taken precisely when they are enabled, in a way that is analogous to some non-standard *urgent/must* semantics for transition guards of hybrid automata.

The hybrid automaton in Fig. 7 models both stable and unstable sliding and is additionally chatter-free, whereas the automaton in Fig. 6 admits chattering runs when the continuous state is at the origin. Of all these operational models, the one employing guard conditions of type II (in Fig. 7) is perhaps the most physically meaningful and faithful to the intended behaviour of the system.

6. OUTLOOK AND RELATED WORK

Having automatic means of computing operational models of systems which can be concisely specified (but whose operational models require an unreasonable amount of effort and care to explicitly write down manually) is a significant enabling factor. In general, computing adequate hybrid automaton models of systems is highly non-trivial [29]. The examples used in this paper are very simple and are intended to highlight differences between the different models; more interesting examples of PWS lead to automata that are indeed quite formidable. We have implemented our HA synthesis algorithm in Mathematica and are able to generate automata in the format of the verification tool SpaceEx [14].[5]

The hybrid automata we are able to generate can provide suitable models for addressing the problem of verification (e.g. of safety and liveness properties) and benefit from a large and growing number of software tools developed to verify or simulate hybrid systems [14, 24, 5, 15, 43]. Verification technology for hybrid systems has improved tremendously in the last two decades; however, in much of existing work there are significant restrictions on the form of hybrid automata, such as e.g. only allowing linear ODEs to govern continuous evolution, or only allowing a specific class of sets (e.g. polytopes) to act as mode invariants for the states of the automaton. We should note that in this sense the class of systems considered in this paper is very broad because it allows for non-linear continuous dynamics and for arbitrary semi-algebraic sets to act as mode invariants and transition guards.

It is our hope our techniques will in future be applied to modelling and verification of properties in systems with engineering applications that employ variable structure control. We stress, however, that many important questions remain unresolved. For instance, the difficult task of categorizing and classifying the possible kinds of operational models (beyond the three presented) remains to be addressed. Interesting questions as to which of the many possible types of operational semantics for PWS that can be obtained through using techniques described in this paper are "physically meaningful" (and for what phenomena) present many intriguing avenues for future research.

6.1 Related Work

Lygeros et al. studied existence and uniqueness of executions of hybrid automata in [26], giving conditions under which hybrid automata are deterministic and non-blocking. We note that there are important differences in definitions, e.g. the use of semi-open time intervals in [26], such as in

$$\mathrm{Out}(q_i) \equiv \{\boldsymbol{x} \in \mathbb{R}^n \mid \forall\, \varepsilon.\ \exists\, t \in [0, \varepsilon).\ \varphi_t(\boldsymbol{x}) \notin M_i\},$$

where $M_i = \mathrm{Dom}(q_i)$. This differs from definitions used in this paper, e.g.

$$\neg\mathrm{In}_{\mathfrak{f}_i}(M_i) \equiv \{\boldsymbol{x} \in \mathbb{R}^n \mid \forall\, \varepsilon.\ \exists\, t \in (0, \varepsilon).\ \varphi_t(\boldsymbol{x}) \notin M_i\}.$$

REMARK 4. *Similar notions also exist in the ODE literature, e.g. "ingress" and "egress" sets used to state and prove the Ważewski principle ([21, p. 282],[7]).*

The work in [26] is also similar in using Lie derivatives of functions to reason about the transition behaviour; however, the authors consider a special class of hybrid automata in which mode invariants can be characterized by sub-level sets of analytic functions, i.e. $\sigma(\boldsymbol{x}) \geq 0$. The same restriction was used in the work of Johansson et al. [23] and already rules out systems in which mode invariants are given by polytopes. We work under much more general assumptions where the mode invariants are semi-algebraic sets and work with their representations directly. Further investigations of existence and uniqueness of executions of hybrid automata were reported in [27].

[5] An implementation is available from http://www.lix.polytechnique.fr/~ghorbal/EMSOFT17.

7. CONCLUSION

In this paper we presented a methodology for automatically synthesizing hybrid automata from descriptions of piecewise-smooth polynomial systems, i.e. systems of discontinuous ODEs that are polynomial on disjoint semi-algebraic sets forming a partition of the state space. The hybrid automata thus obtained provide operational models of piecewise-smooth systems, which can behave in different ways, depending on certain choices in formulating the conditions on the transition guards. We have described in Sections 4.2, 4.3 three alternative choices that can be exercised in this regard, and which can be thought of as giving different operational meaning (i.e. semantics) to the piecewise-smooth systems. Many more choices are possible and the task of studying and classifying these possibilities presents a very interesting direction for further research.

One of our main aims in this paper was to present a case as to why it is not meaningful to speak of "the hybrid automaton model" of a given piecewise-smooth system without a precise description of how the said hybrid automaton model was created. We argue that a synthesis algorithm, such as that presented in Section 4.2, is needed in order to provide this description.

We believe that correct modelling of piecewise-smooth systems is a problem that is of more than just theoretical interest, since systems of this type occur frequently in control engineering (often in the context of autonomous switching or sliding mode controllers). Their representation as differential equations active inside certain designated regions is deceptively simple and great care needs to be taken when extracting operational models from these simple representations. Our work addressed some of the fundamental difficulties inherent in this task.

Acknowledgements.

The authors would like to thank Dr. Gautam Biswas at Vanderbilt University for a useful discussion of his related earlier research and extend special thanks to the anonymous EMSOFT'17 reviewers for their careful reading, valuable suggestions, and for pointing out some relevant references.

8. REFERENCES

[1] R. Alur, C. Courcoubetis, T. A. Henzinger, and P.-H. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems*, pages 209–229. Springer, 1993.

[2] A. D. Ames, A. Abate, and S. Sastry. Sufficient conditions for the existence of zeno behavior. In *Proceedings of the 44th IEEE Conference on Decision and Control*, pages 696–701, Dec 2005.

[3] S. Basu, R. Pollack, and M.-F. Roy. *Algorithms in Real Algebraic Geometry (Algorithms and Computation in Mathematics)*. Springer, 2 edition, 2006.

[4] B. F. Caviness and J. R. Johnson. *Quantifier Elimination and Cylindrical Algebraic Decomposition*. Springer, 1998.

[5] X. Chen, E. Ábrahám, and S. Sankaranarayanan. Flow*: An analyzer for non-linear hybrid systems. In *Proceedings of the 25th International Conference on Computer Aided Verification*, CAV'13, pages 258–263. Springer, 2013.

[6] G. E. Collins. *Quantifier elimination for real closed fields by cylindrical algebraic decompostion*, volume 33 of *Lecture Notes in Computer Science*, pages 134–183. Springer, 1975.

[7] C. C. Conley. *Isolated invariant sets and the Morse index*. Conference Board of the Mathematical Sciences, 1978.

[8] J. Cortés. Discontinuous dynamical systems: A tutorial on solutions, non-smooth analysis and stability. *IEEE Control Systems*, 28(3):36–73, 2008.

[9] D. Cox, J. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra*. Springer, 2010.

[10] J. H. Davenport and J. Heintz. Real quantifier elimination is doubly exponential. *J. Symb. Comput.*, 5(1-2):29–35, Feb. 1988.

[11] J. M. Davoren and A. Nerode. Logics for hybrid systems. *Proc. IEEE*, 88(7):985–1010, 2000.

[12] M. Egerstedt. Behavior based robotics using hybrid automata. In *Proceedings of the Third International Workshop on Hybrid Systems: Computation and Control*, HSCC '00, pages 103–116. Springer, 2000.

[13] G. Frehse. *An Introduction to Hybrid Automata, Numerical Simulation and Reachability Analysis*, pages 50–81. Springer, 2015.

[14] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler. Spaceex: Scalable verification of hybrid systems. In *Proceedings of the 23rd International Conference on Computer Aided Verification*, CAV'11, pages 379–395. Springer, 2011.

[15] N. Fulton, S. Mitsch, J.-D. Quesel, M. Völp, and A. Platzer. KeYmaera X: An axiomatic tactical theorem prover for hybrid systems. In *CADE-25: 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings*, volume 9195 of *Lecutre Notes in Computer Science*. Springer, 2015.

[16] K. Ghorbal and A. Platzer. Characterizing algebraic invariants by differential radical invariants. In *Tools and Algorithms for the Construction and Analysis of Systems - 20th International Conference, TACAS 2014. Proceedings*, pages 279–294, 2014.

[17] K. Ghorbal, A. Sogokon, and A. Platzer. A hierarchy of proof rules for checking positive invariance of algebraic and semi-algebraic sets. *Computer Languages, Systems & Structures*, 47:19–43, 2017.

[18] R. Goebel, R. G. Sanfelice, and A. R. Teel. Hybrid dynamical systems. *IEEE Control Systems*, 29(2):28–93, 2009.

[19] O. Hájek. Discontinuous differential equations I. *Journal of Differential Equations*, 32(2):149 – 170, 1979.

[20] J. K. Hale and J. P. LaSalle. Differential equations: Linearity vs. nonlinearity. *SIAM Review*, 5(3):249–272, July 1963.

[21] P. Hartman. *Ordinary Differential Equations*. John Wiley & Sons, Inc., New York, 1964.

[22] T. A. Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science*, LICS '96, pages 278–292.

IEEE Computer Society, 1996.

[23] K. H. Johansson, M. Egerstedt, J. Lygeros, and S. Sastry. On the regularization of zeno hybrid automata. *Systems & Control Letters*, 38(3):141–150, 1999.

[24] S. Kong, S. Gao, W. Chen, and E. M. Clarke. dReach: δ-reachability analysis for hybrid systems. In *Tools and Algorithms for the Construction and Analysis of Systems - 21st International Conference, TACAS 2015. Proceedings*, pages 200–205. Springer, 2015.

[25] J. Liu, N. Zhan, and H. Zhao. Computing semi-algebraic invariants for polynomial dynamical systems. In *Proceedings of the Ninth ACM International Conference on Embedded Software*, EMSOFT '11, pages 97–106. ACM, 2011.

[26] J. Lygeros, K. H. Johansson, S. Sastry, and M. Egerstedt. On the existence of executions of hybrid automata. In *the 38th IEEE Conference on Decision and Control, Phoenix, AZ*, pages 2249–2254. IEEE, 1999.

[27] J. Lygeros, K. H. Johansson, S. N. Simić, J. Zhang, and S. S. Sastry. Dynamical properties of hybrid automata. *IEEE Transactions on Automatic Control*, 48(1):2–17, Jan 2003.

[28] B. Mishra. *Algorithmic Algebra*. Texts and Monographs in Computer Science. Springer, 1993.

[29] P. J. Mosterman. Mode transition behavior in hybrid dynamic systems. In S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, editors, *Proc. of the 2003 Winter Simulation Conference*, pages 623–631, Dec. 2003.

[30] P. J. Mosterman and G. Biswas. A theory of discontinuities in physical system models. *Journal of the Franklin Institute*, 335(3):401–439, 1998.

[31] P. J. Mosterman, F. Zhao, and G. Biswas. An ontology for transitions in physical dynamic systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence and Tenth Innovative Applications of Artificial Intelligence Conference, AAAI 98, IAAI 98, July 26-30, 1998, Madison, Wisconsin, USA.*, pages 219–224, 1998.

[32] E. M. Navarro-López and R. Carter. Hybrid automata: an insight into the discrete abstraction of discontinuous systems. *International Journal of Systems Science*, 42(11):1883–1898, 2011.

[33] D. Novikov and S. Yakovenko. Trajectories of polynomial vector fields and ascending chains of polynomial ideals. In *Annales de l'institut Fourier*, volume 49, pages 563–609, 1999.

[34] A. Platzer. Differential dynamic logic for hybrid systems. *J. Autom. Reas.*, 41(2):143–189, 2008.

[35] A. Platzer. *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Springer, 2010.

[36] R. G. Sanfelice, R. Goebel, and A. R. Teel. Generalized solutions to hybrid dynamical systems. *ESAIM: Control, Optimisation and Calculus of Variations*, 14:699–724, 10 2008.

[37] A. Seidenberg. A new decision method for elementary algebra. *Annals of Mathematics*, pages 365–374, 1954.

[38] A. Tarski. A decision method for elementary algebra and geometry. *Bulletin of the American Mathematical Society*, 59, 1951.

[39] G. Teschl. *Ordinary Differential Equations and Dynamical Systems*, volume 140 of *Graduate Studies in Mathematics*. American Mathematical Society, 2012.

[40] A. Tiwari. Abstractions for hybrid systems. *Formal Methods in System Design*, 32(1):57–83, 2008.

[41] V. I. Utkin. *Sliding Modes in Control and Optimization*. Communications and Control Engineering Series. Springer, 1992.

[42] A. J. Van Der Schaft and H. Schumacher. *An introduction to hybrid dynamical systems*, volume 251 of *Lecture Notes in Control and Information Sciences*. Springer, 2000.

[43] S. Wang, N. Zhan, and L. Zou. *An Improved HHL Prover: An Interactive Theorem Prover for Hybrid Systems*, pages 382–399. Springer, 2015.

[44] H. S. Witsenhausen. A class of hybrid-state continuous-time dynamic systems. *IEEE Transactions on Automatic Control*, 11(2):161–167, Apr 1966.

[45] F. Zhao and V. I. Utkin. Adaptive simulation and control of variable-structure control systems in sliding regimes. *Automatica*, 32(7):1037 – 1042, 1996.

APPENDIX

A. COMPUTING "IN SETS" EXACTLY

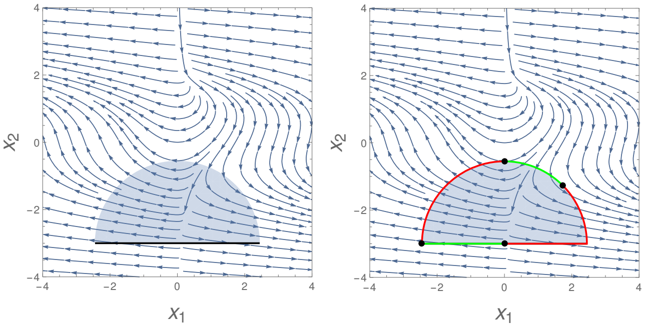
To give an idea of how $\mathrm{In}_{\mathfrak{f}}(S)$ is computed exactly, consider a set S which is given by $p \leq 0$, where p is some polynomial function in the state variables $x_1, \ldots, x_n$ with real coefficients. Firstly, note that each point $\boldsymbol{x}$ in the interior of S, i.e. satisfying $p < 0$ necessarily lies inside $\mathrm{In}_{\mathfrak{f}}(p \leq 0)$ because motion within the interior is always possible within some open neighbourhood. The set $p < 0$ thus provides the first under-approximation of the set $\mathrm{In}_{\mathfrak{f}}(p \leq 0)$. We now refine this under-approximation by adding some of the states satisfying $p = 0$, for which a *sufficient* (but not necessary) condition for membership in $\mathrm{In}_{\mathfrak{f}}(p \leq 0)$ is that of satisfying the inequality $\mathfrak{L}_{\mathfrak{f}}(p) < 0$. This is intuitive because the rate of change of p at such a state is negative and therefore the system will immediately evolve into the set satisfying $p < 0$. However, for states satisfying $p = 0$ and $\mathfrak{L}_{\mathfrak{f}}(p) = 0$, one needs to check that the second-order Lie derivative is negative (i.e. $\mathfrak{L}_{\mathfrak{f}}^2(p) < 0$) in order to conclude their membership in $\mathrm{In}_{\mathfrak{f}}(p \leq 0)$, and so on for higher-order Lie derivatives. Intuitively, these cases correspond to situations where "the velocity is zero, but the acceleration is negative", etc., which likewise ensures that the system cannot evolve into a state satisfying $p > 0$ (i.e. the complement of $p \leq 0$) immediately afterwards. The set $\mathrm{In}_{\mathfrak{f}}(p \leq 0)$ is then constructed as follows:

$$\begin{aligned}
\mathrm{In}_{\mathfrak{f}}(p \leq 0) \equiv\ & p < 0 \\
& \vee (p = 0 \wedge \mathfrak{L}_{\mathfrak{f}}(p) < 0) \\
& \vee (p = 0 \wedge \mathfrak{L}_{\mathfrak{f}}(p) = 0 \wedge \mathfrak{L}_{\mathfrak{f}}^2(p) < 0) \\
& \vdots \\
& \vee (p = 0 \wedge \mathfrak{L}_{\mathfrak{f}}(p) = 0 \wedge \cdots \wedge \mathfrak{L}_{\mathfrak{f}}^k(p) \leq 0)
\end{aligned}$$

The fact that the number k is finite and can be computed is a consequence of Hilbert's basis theorem and the ascending chain property of Noetherian rings (see e.g. [28, Sec. 2.3.2]). These fundamental results guarantee that one is always able to find a $k \in \mathbb{N}$ such that the ideal membership $\mathfrak{L}_{\mathfrak{f}}^{K}(p) \in \langle p, \mathfrak{L}_{\mathfrak{f}}(p), \ldots, \mathfrak{L}_{\mathfrak{f}}^{k}(p)\rangle$ [6] holds for all $K \geq k$. This property is equivalent to the statement that for each $K \geq k$ the following equality holds: $\mathfrak{L}_{\mathfrak{f}}^{K}(p) = \alpha_0 p + \alpha_1 \mathfrak{L}_{\mathfrak{f}}(p) + \cdots + \alpha_k \mathfrak{L}_{\mathfrak{f}}^{k}(p)$, where the coefficients $\alpha_0, \alpha_1, \ldots, \alpha_k$ are some polynomials in the ring $\mathbb{R}[x_1, \ldots, x_n]$. Thus, whenever $p = \mathfrak{L}_{\mathfrak{f}}(p) = \cdots = \mathfrak{L}_{\mathfrak{f}}^{k}(p) = 0$ holds, one necessarily has $\mathfrak{L}_{\mathfrak{f}}^{K}(p) = 0$ for all $K \geq 0$, and thus it is impossible to grow the under-approximation of $\mathrm{In}_{\mathfrak{f}}(p \leq 0)$ by adding any more disjuncts of the form $p = 0 \wedge \mathfrak{L}_{\mathfrak{f}}(p) = 0 \wedge \cdots \wedge \mathfrak{L}_{\mathfrak{f}}^{k}(p) = 0 \wedge \cdots \wedge \mathfrak{L}_{\mathfrak{f}}^{K}(p) < 0$ for any $K > k$ and the construction is therefore complete. In practice, the number k is computed using Gröbner bases (e.g. see [9, Chap. 2]).

B. ENTER, EXIT AND BOUNCE SETS

Consider a semi-algebraic set described by the formula $S \equiv x_1^2 + (x_2+3)^2 < 6 \wedge -3 \leq x_2$ and let the dynamics of the system, $\dot{\boldsymbol{x}} = \mathfrak{f}(\boldsymbol{x})$, be given by the system of polynomial ODEs: $\dot{x}_1 = x_1 x_2^2 - 1, \dot{x}_2 = -x_1$. Fig. 9a shows the set S along with some of the trajectories of the system. The set of "entering states", given by



(a) Semi-algebraic set $S \subset \mathbb{R}^2$ (b) $\mathrm{Enter}_{\mathfrak{f}}(S)$ and $\mathrm{Exit}_{\mathfrak{f}}(S)$

Figure 9: Semi-algebraic set, along with its "entry" states (in green) and "exit" states (in red).

$$\mathrm{Enter}_{\mathfrak{f}}(S) = \big(x_2 + 3 = 0 \wedge x_1 < 0 \wedge x_1^2 + x_2^2 + 6x_2 + 3 < 0\big)$$
$$\vee\ (x_2 + 3 > 0 \wedge x_1^2 + x_2^2 + 6x_2 + 3 = 0 \wedge x_1^2 x_2^2 < x_1\,(x_2 + 4)),$$

is shown in green in Fig. 9b, and

$$\mathrm{Exit}_{\mathfrak{f}}(S) = \Big(0 < x_1 \leq \sqrt{6} \wedge x_2 + 3 = 0\Big) \vee \big(x_2 + 3 > 0$$
$$\wedge\ x_1^2 + x_2\,(x_2 + 6) + 3 = 0 \wedge x_1^2 x_2^2 > x_1\,(x_2 + 4)\,\big)$$

is shown in red. Note that these two sets need not necessarily include all the points on the boundary of S. The black points in Fig. 9b represent states on the boundary which are neither in $\mathrm{Enter}_{\mathfrak{f}}(S)$ nor $\mathrm{Exit}_{\mathfrak{f}}(S)$. In particular, $\mathrm{Bounce}_{\mathfrak{f}}(S)$ includes the point at the centre of the semi-circle, i.e. $x_1 = 0 \wedge x_2 = -3$, whereas the remaining three points in the figure belong to $\mathrm{Bounce}_{\mathfrak{f}}(\neg S)$.

[6] i.e. $\mathfrak{L}_{\mathfrak{f}}^{K}(p)$ is in the ideal generated by the finite set of polynomials $\{p, \mathfrak{L}_{\mathfrak{f}}(p), \ldots, \mathfrak{L}_{\mathfrak{f}}^{k}(p)\}$

Verifying safety and persistence properties of hybrid systems using flowpipes and continuous invariants *

Andrew Sogokon[1], Paul B. Jackson[2], and Taylor T. Johnson[1]

[1] Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN, USA
{andrew.sogokon|taylor.johnson}@vanderbilt.edu
[2] Laboratory for Foundations of Computer Science, University of Edinburgh, Scotland, UK
Paul.Jackson@ed.ac.uk

Abstract We propose a method for verifying persistence of nonlinear hybrid systems. Given some system and an initial set of states, the method can guarantee that system trajectories always eventually evolve into some specified target subset of the states of one of the discrete modes of the system, and always remain within this target region. The method also computes a time-bound within which the target region is always reached. The approach combines flow-pipe computation with deductive reasoning about invariants and is more general than each technique alone. We illustrate the method with a case study concerning showing that potentially destructive stick-slip oscillations of an oil-well drill eventually die away for a certain choice of drill control parameters. The case study demonstrates how just using flow-pipes or just reasoning about invariants alone can be insufficient. The case study also nicely shows the richness of systems that the method can handle: the case study features a mode with non-polynomial (nonlinear) ODEs and we manage to prove the persistence property with the aid of an automatic prover specifically designed for handling transcendental functions.

1 Introduction

Hybrid systems combine discrete and continuous behaviour and provide a very general framework for modelling and analyzing the behaviour of systems such as those implemented in modern embedded control software. Although a number of tools and methods have been developed for verifying properties of hybrid systems, most are geared towards proving bounded-time safety properties, often employing set reachability computations based on constructing over-approximating enclosures of the reachable states of ordinary differential equations (e.g. [7,14,13,21]). Methods capable of proving unbounded-time safety properties often rely (explicitly or otherwise) on constructing *continuous invariants* (e.g. [42,25], and referred to in short as *invariants*). Such invariants may be thought of as a generalization of *positively invariant sets* (see e.g. [5]) and which are analogous to inductive invariants used in computer science to reason about the correctness of discrete programs using Hoare logic.

* This material is based upon work supported by the UK Engineering and Physical Sciences Research Council under grants EPSRC EP/I010335/1 and EP/J001058/1, the National Science Foundation (NSF) under grant numbers CNS 1464311 and CCF 1527398, the Air Force Research Laboratory (AFRL) through contract number FA8750-15-1-0105, and the Air Force Office of Scientific Research (AFOSR) under contract number FA9550-15-1-0258.

We argue in this paper that a combined approach employing bounded time reachability analysis and reasoning about invariants can be effective in proving *persistence* and *safety* properties in non-polynomial (nonlinear) hybrid systems. We illustrate the combined approach using a detailed case study with non-polynomial ODEs for which neither approach individually was sufficient to establish the desired safety and persistence properties.

Methods for bounded time safety verification cannot in general be applied to prove safety for all time and their accuracy tends to degrade for large time bounds, especially for nonlinear systems. Verification using invariants, while a powerful technique that can prove strong properties about nonlinear systems, relies on the ability to find invariants that are sufficient for proving the unbounded time safety property. In practice, many invariants for the system can be found which fall short of this requirement, often for the simple reason that they do not include all the initial states of the system. We show how a combined approach employing both verification methods can, in some cases, address these limitations.

Contributions.

In this paper we (**I**) show that bounded time safety verification based on flowpipe construction can be naturally combined with invariants to verify persistence and unbounded time safety properties, addressing some of the limitations of each verification method when considered in isolation. (**II**) To illustrate the approach, we consider a simplified torsional model of a conventional oil well drill string that has been the subject of numerous studies by Navarro-López et al. [34]. (**III**) We discuss some of the challenges that currently stand in the way of fully automatic verification using this approach. Additionally, we provide a readable overview of the methods employed in the verification process and the obstacles that present themselves when these methods are applied in practice.

2 Safety and Persistence for Hybrid Automata

2.1 Preliminaries

A number of formalisms exist for specifying hybrid systems. The most popular framework at present is that of hybrid automata [3,19], which are essentially discrete transition systems in which each discrete state represents an operating mode inside which the system evolves continuously according to an ODE under some evolution constraint. Additionally, transition guards and reset maps are used to specify the discrete transition behaviour (i.e. switching) between the operating modes. A sketch of the syntax and semantics of hybrid automata is as follows.

Definition 1 (Hybrid automaton [26]). *Formally, a hybrid automaton is given by* $(Q, Var, \boldsymbol{f}, Init, Inv, T, G, R)$, *where*

- $Q = \{q_0, q_1, \ldots, q_k\}$ *is a* finite *set of discrete states (modes),*
- $Var = \{x_1, x_2, \ldots, x_n\}$ *is a* finite *set of continuous variables,*
- $f : Q \times \mathbb{R}^n \to \mathbb{R}^n$ *gives the vector field defining continuous evolution inside each mode,*
- $Init \subset Q \times \mathbb{R}^n$ *is the set of initial states,*

- $\mathit{Inv} : Q \to 2^{\mathbb{R}^n}$ *gives the* mode invariants *constraining evolution for every discrete state,*
- $T \subseteq Q \times Q$ *is the transition relation,*
- $G : T \to 2^{\mathbb{R}^n}$ *gives the guard conditions for enabling transitions,*
- $R : T \to 2^{\mathbb{R}^n \times \mathbb{R}^n}$ *gives the reset map.*

A *hybrid state* of the automaton is of the form $(q, \boldsymbol{x}) \in Q \times \mathbb{R}^n$. A *hybrid time trajectory* is a sequence (which may be finite or infinite) of intervals $\tau = \{I_i\}_{i=0}^{N}$, for which $I_i = [\tau_i, \tau'_i]$ for all $i < N$ and $\tau_i \leq \tau'_i = \tau_{i+1}$ for all i. If the sequence is finite, then either $I_N = [\tau_N, \tau'_N]$ or $I_N = [\tau_N, \tau'_N)$. Intuitively, one may think of τ_i as the times at which discrete transitions occur. An *execution* (or a *run* or *trajectory*) of a hybrid automaton defined to be $(\tau, q, \varphi^i_t(\boldsymbol{x}))$, where τ is a hybrid time trajectory, $q : \langle\tau\rangle \to Q$ (where $\langle\tau\rangle$ is defined to be the set $\{0, 1, \ldots, N\}$ if τ is finite and $\{0, 1, \ldots\}$ otherwise) and $\varphi^i_t(\boldsymbol{x})$ is a collection of diffeomorphisms $\varphi^i_t(\boldsymbol{x}) : I_i \to \mathbb{R}^n$ such that $(q(0), \varphi^0_0(\boldsymbol{x})) \in \mathit{Init}$, for all $t \in [\tau_i, \tau'_i)$ $\dot{\boldsymbol{x}} = f(q(i), \varphi^i_t(\boldsymbol{x}))$ and $\varphi^i_t(\boldsymbol{x}) \in \mathit{Inv}(i)$. For all $i \in \langle\tau\rangle \setminus \{N\}$ it is also required that transitions respect the guards and reset maps, i.e. $e = (q(i), q(i+1)) \in T$, $\varphi^i_{\tau'_i}(\boldsymbol{x}) \in G(e)$ and $(\varphi^i_{\tau'_i}(\boldsymbol{x}), \varphi^{i+1}_{\tau_{i+1}}(\boldsymbol{x})) \in R(e)$.

We consider MTL[3] formulas satisfied by trajectories. The satisfaction relation is of form $\rho \models^p \phi$, read as "*trajectory ρ at position p satisfies temporal logic formula ϕ*", where positions on a trajectory are identified by pairs of form (i, t) where $i \leq N$ and time $t \in I_t$. We use the MTL modality $\Box_I \phi$ which states that formula ϕ always holds in time interval I in the future. Formally, this can be defined as $\rho \models^p \Box_I \phi \;\equiv\; \forall p' \geq p \text{ s.t. } (p'.2 - p.2) \in I.\, \rho \models^{p'} \phi$, where $(i', t') \geq (i, t) \equiv i' > i \vee (i' = i \wedge t' \geq t)$. Similarly we can define the modality $\Diamond_I \phi$ which states that formula ϕ eventually holds at some time in the time interval I in the future. An MTL formula is valid for a given hybrid automaton if it is satisfied by all trajectories of that automaton starting at position $(0, 0)$. For clarity when writing MTL formulas, we assume trajectories are not restricted to start in Init states and instead introduce Init predicates into the formulas when we want restrictions.

Alternative formalisms for hybrid systems, such as *hybrid programs* [41], enjoy the property of having a compositional semantics and can be used to verify properties of systems by verifying properties of their parts in a theorem prover [44,15]. Other formal modelling frameworks for hybrid systems, such as *Hybrid CSP* [24], have also found application in theorem provers [60,62].

2.2 Bounded Time Safety and Eventuality

The *bounded-time safety verification problem* (with some finite time bound $t > 0$) is concerned with establishing that given an initial set of states $\mathrm{Init} \subseteq Q \times \mathbb{R}^n$ and a set of safe states $\mathrm{Safe} \subseteq Q \times \mathbb{R}^n$, the state of the system may not leave Safe within time t along any valid trajectory τ of the system. In the absence of closed-form solutions to the ODEs, this property may be established by verified integration, i.e. by computing successive over-approximating enclosures (known as *flowpipes*) of the reachable states in discrete time steps. Bounded-time reachability analysis can be extended to full hybrid systems by also computing/over-approximating the discrete reachable states (up to some finite bound on the number of discrete transitions).

[3] Metric Temporal Logic; see e.g. [22].

A number of bounded-time verification tools for hybrid systems have been developed based on verified integration using interval enclosures. For instance, *iSAT-ODE*, a verification tool for hybrid systems developed by Eggers et al. [13] relies on the verified integration tool *VNODE-LP* by Nedialkov [37] for computing the enclosures. Other examples include *dReach*, a reachability analysis tool for hybrid systems developed by Kong et al. [21], which uses the *CAPD* library [1]. Over-approximating enclosures can in practice be very precise for small time horizons, but tend to become conservative when the time bound is large (due to the so-called *wrapping effect*, which is a problem caused by the successive build-up of over-approximation errors that arises in interval-based methods; see e.g. [38].) An alternative verified integration method using *Taylor models* was introduced by Makino and Berz (see [4,38]) and can address some of these drawbacks, often providing tighter enclosures of the reachable set. Implementations of the method have been reported in *COSY INFINITY*, a scientific computing tool by Makino and Berz [29]; *VSPODE*, a tool for computing validated solutions to parametric ODEs by Lin and Stadtherr [23]; and in *Flow**, a bounded-time verification for hybrid systems developed by Chen et al. [7].

Because flowpipes provide an over-approximation of the reachable states at a given time, verified integration using flowpipes can also be used to reason about *liveness* properties such as *eventuality*, i.e. when a system is guaranteed to eventually enter some *target set* having started off at some point in an initial set. The bounded-time safety and eventuality properties may be more concisely expressed by using MTL notation, i.e. by writing $\mathrm{Init} \to \Box_{[0,t]}\,\mathrm{Safe}$, and $\mathrm{Init} \to \Diamond_{[0,t]}\,\mathrm{Target}$, where Init describes the initial set of states, $\mathrm{Safe} \subseteq Q \times \mathbb{R}^n$ is the set of safe states and $\mathrm{Target} \subseteq Q \times \mathbb{R}^n$ is the target region which is to be eventually attained.

Remark 2. The bounded time eventuality properties we consider in this paper are more restrictive than the general (unbounded time) case. For instance, consider a continuous 2-dimensional system governed by $\dot{x}_1 = x_2, \dot{x}_2 = 0$ and confined to evolve in the region where $x_2 > 0$. If one starts this system inside a state where $x_1 = 0$, it will eventually evolve into a state where $x_1 = 1$ by following the solution, however one may not put a finite bound on the time for this to happen. Thus, while $x_1 = 0 \to \Diamond_{[0,\infty)}\, x_1 = 1$ is true for this system the bounded time eventuality property $x_1 = 0 \to \Diamond_{[0,t]}\, x_1 = 1$, will not hold for any finite $t > 0$.

2.3 Unbounded Time Safety

A safety property for unbounded time may be more concisely expressed using an MTL formula:

$$\mathrm{Init} \to \Box_{[0,\infty)}\,\mathrm{Safe}.$$

A proof of such a safety assertion is most commonly achieved by finding an appropriate *invariant*, $I \subseteq Q \times \mathbb{R}^n$, which contains no unsafe states (i.e. $I \subseteq \mathrm{Safe}$) and such that the state of the system may not escape from I into an unsafe state along any valid trajectory of the system. Invariance is a special kind of safety assertion and may be written as $\mathrm{I} \to \Box_{[0,\infty)}\,\mathrm{I}$. A number of techniques have been developed for proving invariance properties for continuous systems without the need to compute solutions to the ODEs [49,41,58,25,17,53].

2.4 Combining Unbounded Time Safety with Eventuality to Prove Persistence

In linear temporal logic, a *persistence* property states that a formula is 'eventually always' true. For instance, using persistence one may express the property that a system starting in any initial state always eventually reaches some target set and then always stays within this set. Using MTL notation, we can write this as:

$$\mathrm{Init} \to \Diamond_{[0,\infty)} \Box_{[0,\infty)} \mathrm{Target}.$$

Persistence properties generalize the concept of stability. With stability one is concerned with showing that the state of a system always converges to some particular equilibrium point. With persistence, one only requires that the system state eventually becomes always trapped within some set of states.

In this paper we are concerned with a slightly stronger form of persistence, where one ensures that the target set is always reached within some specified time t:

$$\mathrm{Init} \to \Diamond_{[0,t]} \Box_{[0,\infty)} \mathrm{Target}.$$

We observe that a way of proving this is to find a set $I \subseteq \mathrm{Target}$ such that:

1. $\mathrm{Init} \to \Diamond_{[0,t]} I$ holds, and
2. I is an invariant for the system.

This fact can be stated more formally as a rule of inference:

$$\text{(Persistence)}\ \frac{\mathrm{Init} \to \Diamond_{[0,t]} I \qquad I \to \Box_{[0,\infty)} I \qquad I \to \mathrm{Target}}{\mathrm{Init} \to \Diamond_{[0,t]} \Box_{[0,\infty)} \mathrm{Target}}\ .$$

Previous Sections 2.2 and 2.3 respectively surveyed how the eventuality premise $\mathrm{Init} \to \Diamond_{[0,t]} I$ and invariant premise $I \to \Box_{[0,\infty)} I$ can be established by a variety of automated techniques. In Section 5 we explore automation challenges further and remark on ongoing work addressing how to automatically generate suitable invariants I.

2.5 Using Persistence to Prove Safety

Finding appropriate invariants to prove unbounded time safety as explained above in Section 2.3 can in practice be very difficult. It might be the case that invariants $I \subseteq \mathrm{Safe}$ for the system can be found, but also ensuring that $\mathrm{Init} \subseteq I$ is infeasible. Nevertheless it might be the case that one of these invariants I is always eventually reached by trajectories starting in Init and all those trajectories are contained within Safe. In such cases, Safe is indeed a safety property of the system when starting from any point in Init. More precisely, if one can find an invariant I as explained above in Section 2.4 to show the persistence property: $\mathrm{Init} \to \Diamond_{[0,t]} \Box_{[0,\infty)} \mathrm{Safe}$, and further one can show for the same time bound t that: $\mathrm{Init} \to \Box_{[0,t]} \mathrm{Safe}$, then one has: $\mathrm{Init} \to \Box_{[0,\infty)} \mathrm{Safe}$. As a result, one may potentially utilize invariants that were by themselves insufficient for proving the safety property.

Remark 3. The problem of showing that a state satisfying $\Box_{[0,\infty)} \mathrm{Safe}$ is reached in finite time t, while ensuring that the formula $\Box_{[0,t]} \mathrm{Safe}$ also holds (i.e. states satisfying $\neg\mathrm{Safe}$ are avoided up to time t) is sometimes called a *reach-avoid problem* [61].

Even if one's goal is to establish bounded-time rather than unbounded-time safety properties, this inference scheme could still be of use, as it could significantly reduce the time bound t needed for bounded time reachability analysis. In practice, successive over-approximation of the reachable states using flowpipes tends to become conservative for large values of t. In highly non-linear systems one can realistically expect to compute flowpipes only for very modest time bounds (e.g. in chaotic systems flowpipes are guaranteed to 'blow up', but invariants may still sometimes be found). Instead, it may in some cases be possible to prove the safety property by computing flowpipes up to some small time bound, after which the system can be shown to be inside an invariant that implies the safety property for all times thereafter.

3 An example persistence verification problem

Stick-slip oscillations are commonly encountered in mechanical engineering in the context of modelling the effects of dynamic friction. Informally, the phenomenon manifests itself in the system becoming "stuck" and "unstuck" repeatedly, which results in unsteady "jerky" motions. In engineering practice, stick-slip oscillations can often degrade performance and cause failures when operating expensive machinery [36]. Although the problem of demonstrating absence of stick-slip oscillations in a system is primarily motivated by safety considerations, it would be misleading to call this a *safety verification problem*. Instead, the problem may broadly be described as that of demonstrating that the system (in finite time) enters a state in which no stick-slip motion is possible and remains there indefinitely. Using MTL one may write:

$$\text{Init} \to \Diamond_{[0,t]} \Box_{[0,\infty)} \text{Steady},$$

where Steady describes the states in which harmful oscillations cannot occur. The formula may informally be read as saying that "from any initial configuration, the system will eventually evolve within time t into a state region where it is always steady".

As an example of a system in which eventual absence of stick-slip oscillations is important, we consider a well-studied [34] model of a simplified conventional oil well drill string. The system can be characterized in terms of the following variables: φ_r, the angular displacement of the top rotary system; φ_b, the angular displacement of the drilling bit; $\dot{\varphi}_r$, the angular velocity of the top rotary system; and $\dot{\varphi}_b$, the angular velocity of the drilling bit. The continuous state of the system $\boldsymbol{x}(t) \in \mathbb{R}^3$ can be described in terms of these variables, i.e. $\boldsymbol{x}(t) = (\dot{\varphi}_r,\ \varphi_r - \varphi_b,\ \dot{\varphi}_b)^T$. The system has two control parameters: W_{ob} giving the weight applied on the drilling bit, and $u = T_m$ giving the surface motor torque. The dynamics is governed a non-linear system of ODEs $\dot{\boldsymbol{x}} = f(\boldsymbol{x})$, given by:

$$\dot{x}_1 = \frac{1}{J_r}\Big(-(c_t + c_r)x_1 - k_t x_2 + c_t x_3 + u\Big), \tag{1}$$

$$\dot{x}_2 = x_1 - x_3, \tag{2}$$

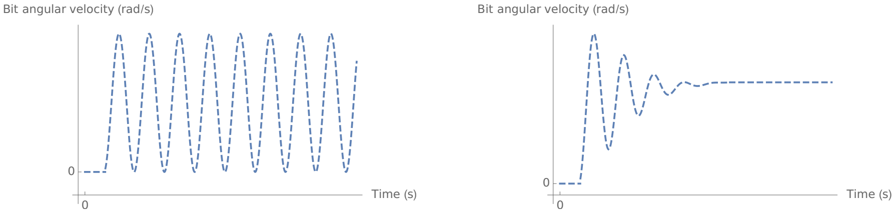
$$\dot{x}_3 = \frac{1}{J_b}\Big(c_t x_1 + k_t x_2 - (c_t + c_b)x_3 - T_{f_b}(x_3)\Big). \tag{3}$$

The term $T_{f_b}(x_3)$ denotes the friction modelling the bit-rock contact and is responsible for the non-polynomial non-linearity. It is given by

$$W_{ob}R_b\Big(\mu_{c_b} + (\mu_{s_b} - \mu_{c_b})e^{-\frac{\gamma_b}{\nu_f}|x_3|}\Big)\mathrm{sgn}(x_3),$$

where $\mathrm{sgn}(x_3) = \frac{x_3}{|x_3|}$ if $x_3 \neq 0$ and $\mathrm{sgn}(x_3) \in [-1, 1]$ if $x_3 = 0$. Constants used in the model [34] are as follows: $c_b = 50\,\mathrm{Nms/rad}$, $k_t = 861.5336\ \mathrm{Nm/rad}$, $J_r = 2212\,\mathrm{kg\,m}^2$, $J_b = 471.9698\ \mathrm{kg\,m}^2$, $R_b = 0.155575\,\mathrm{m}$, $c_t = 172.3067\ \mathrm{Nms/rad}$, $c_r = 425\,\mathrm{Nms/rad}$, $\mu_{cb} = 0.5$, $\mu_{sb} = 0.8$, $\gamma_b = 0.9$, $\nu_f = 1\ \mathrm{rad/s}$. Even though at first glance the system looks like a plain continuous system with a single set of differential equations, it is effectively a hybrid system with at least 3 modes, where the drilling bit is: “rotating forward” ($x_3 > 0$), “stopped” ($x_3 = 0$), and “rotating backward” ($x_3 < 0$). A sub-mode of the stopped mode models when the drill bit is stuck. In this sub-mode, the torque components on the drill bit due to c_t, c_b and k_t are insufficient to overcome the static friction $W_{ob}R_b\mu_{c_b}$, and $\mathrm{sgn}(x_3)$ is further constrained so as to ensure $\dot{x}_3 = 0$.

Once the drill is in operation, so-called *stick-slip oscillations* can cause damage when the bit repeatedly becomes stuck and unstuck due to friction in the bottom hole assembly. In the model this behaviour would correspond to the system entering a state where $x_3 = 0$ repeatedly. The objective is to verify the eventual absence of stick-slip oscillations in the system initialised at the origin (i.e. at rest) for some given choice of the control parameters W_{ob} and u. Previous work by Navarro-López and Carter [34] explored modelling the simplified model of the drill as a hybrid automaton and simulated the resulting models in Stateflow and Modelica.



(a) Stick-slip motion (undesirable)

(b) Stabilization (desired behaviour)

Figure 1: Simulations can exhibit stabilization with positive bit angular velocity and stick-slip bit motion.

Simulations, such as those obtained in [34], using different models and control parameters for the drill can suggest stick-slip oscillations or their absence (illustrated in Fig. 1) in a particular model, however the task of verifying their eventual absence cannot be adequately addressed with simulation alone. In practice however, simulation is incredibly useful in providing some degree of confidence in the overall result, which is very important to know before attempting verification.

A simulation of the system with a concrete choice for the control parameters $W_{ob} = 50,000\ \mathrm{N}$ and $u = 6,000\ \mathrm{Nm}$, shown as a trajectory in the 3-dimensional state space in Fig 3a, suggests that the system does not exhibit stick-slip oscillations, because the trajectory is observed to start at the origin, escape the surface ($x_3 = 0$)[4] and stabilize around a point where the angular velocity of the drilling bit is positive ($x_3 > 0$).

4 Verifying Persistence

The property of interest, i.e. the eventual absence of stick-slip oscillation that we observe in the simulation, may be phrased as the following formula in metric temporal logic: $x_1 = 0 \wedge x_2 = 0 \wedge x_3 = 0 \rightarrow \Diamond_{[0,t]} \, \Box_{[0,\infty)} \, x_3 > 0$, which informally asserts that the system initialised at the origin will *eventually* (diamond modality) enter a state where it is *always* (box modality) the case that $x_3 > 0$. In the following sections we describe a method for proving this assertion. Following our approach, we break the problem down into the following two sub-problems:

1. Finding an appropriate invariant I in which the property $\Box_{[0,t]} \, x_3 > 0$ holds. For this we employ continuous/positive invariants, discussed in the next section.
2. Proving that the system reaches a state in the set I in finite time when initialised at the origin, i.e. $x_1 = 0 \wedge x_2 = 0 \wedge x_3 = 0 \rightarrow \Diamond_{[0,t]} \, I$. [5]

4.1 Continuous Invariant

Finding continuous invariants that are sufficient to guarantee a given property is in practice remarkably difficult. Methods for automatic continuous invariant generation have been reported by numerous authors [49,59,18,53,52,25,63,16,30,54], but in practice often result in "coarse" invariants that cannot be used to prove the property of interest, or require an unreasonable amount of time due to their reliance on expensive real quantifier elimination algorithms.

Stability analysis (involving a linearisation; see [56] for details) can be used to suggest a polynomial function $V : \mathbb{R}^n \rightarrow \mathbb{R}$, given by

$$\begin{aligned} V(\boldsymbol{x}) = {} & 50599.6 - 14235.7x_1 + 1234.22x_1^2 - 4351.43x_2 + 342.329x_1x_2 \\ & + 288.032x_2^2 - 3865.81x_3 + 367.657x_1x_3 + 18.2594x_2x_3 + 241.37x_3^2, \end{aligned}$$

for which we can reasonably conjecture that $V(\boldsymbol{x}) \leq 1400$ defines a positively invariant set under the flow of our non-linear system. Geometrically, this represents an ellipsoid that lies above the surface defined by $x_3 = 0$ in the state space (see Fig. 3b). In order to prove the invariance property, it is sufficient to show that the following holds:[6]

$$\forall \, \boldsymbol{x} \in \mathbb{R}^3. \; V(\boldsymbol{x}) = 1400 \rightarrow \nabla V \cdot f(\boldsymbol{x}) < 0. \tag{4}$$

Unfortunately, in the presence of non-polynomial terms[7] a first order sentence will in general not belong to a decidable theory [51], although there has recently been progress in broadening the scope of the popular CAD algorithm [9] for real quantifier elimination to work with restricted classes of non-polynomial problems [57].

[4] The system exhibits *sliding behaviour* on a portion of this surface known as the *sliding set*. See [34].

[5] Files for the case study are available online. `http://www.verivital.com/nfm2017`

[6] Here ∇ denotes the *gradient* of V, i.e. the vector of partial derivatives $(\frac{\partial V}{\partial x_1}, \ldots, \frac{\partial V}{\partial x_n})$.

[7] E.g. those featured in the right-hand side of the ODE, i.e. $f(\boldsymbol{x})$.

In practice, this conjecture is easily proved in under 5 seconds using MetiTarski, an automatic theorem prover, developed by L.C. Paulson and co-workers at the University of Cambridge, designed specifically for proving universally quantified first order conjectures featuring transcendental functions (such as $\sin$,$\cos$, $\ln$, $\exp$, etc.) The interested reader may find more details about the MetiTarski system in [2,40].

Remark 4. Although Wolfram's *Mathematica* 10 computer algebra system also provides some functionality for proving first-order conjectures featuring non-polynomial expressions using its `Reduce[]` function, we were unable (on our system[8]) to prove conjecture (4) this way after over an hour of computation, after which the Mathematica kernel crashed.

The automatic proof of conjecture (4) obtained using MetiTarski (provided we trust the system) establishes that $V(\boldsymbol{x}) \leq 1400$ defines a positively invariant set, and thus we are guaranteed that solutions initialised inside this set remain there at all future times. In order to be certain that no outgoing discrete transitions of the hybrid system are possible when the system is evolving inside $V(\boldsymbol{x}) \leq 1400$, we further require a proof of the following conjecture featuring only polynomial terms:

$$\forall\, \boldsymbol{x} \in \mathbb{R}^3.\ V(\boldsymbol{x}) \leq 1400 \rightarrow x_3 > 0. \tag{5}$$

An automatic proof of this conjecture may be obtained using an implementation of a decision procedure for first-order real arithmetic.

4.2 Verified Integration

In order to show that the system does indeed enter the positively invariant ellipsoid $V(\boldsymbol{x}) \leq 1400$ in finite time, it is not sufficient to observe this in a simulation (as in Fig. 3b), which is why we use a tool employing *verified integration* based on Taylor models. *Flow** (implemented by Chen et al. [7]) is a bounded-time safety verification tool for hybrid systems that computes Taylor models to analyze continuous reachability. The tool works by computing successive over-approximations (flowpipes) of the reachable set of the system, which are internally represented using Taylor models (but which may in turn be over-approximated by a bounding hyper-box and easily rendered).

Fig. 2a shows the bounding boxes of solution enclosures computed from the point initial condition at the origin using Flow* with adaptive time steps and Taylor models of order 13, a time bound of 12.7 and the same control parameters used in the simulation (i.e. $u = 6,000$ Nm, $W_{ob} = 50,000$ N). We observe that once solutions escape to the region where $x_3 > 0$, they maintain a positive x_3 component for the duration of the time bound.

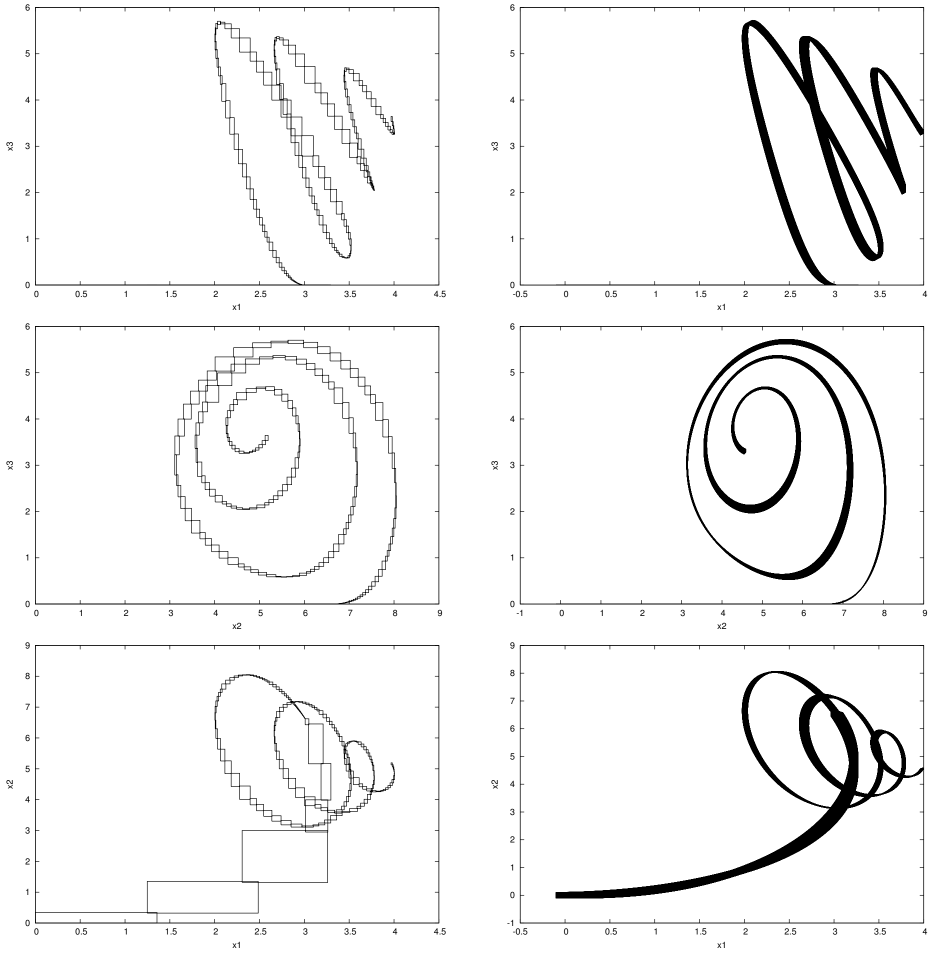
The last flowpipe computed by Flow* for this problem can be bounded inside the hyper-rectangle $\mathrm{BoundBox}$ characterized by the formula

$$\mathrm{BoundBox} \equiv \frac{39}{10} \leq x_1 \leq 4 \wedge \frac{51}{10} \leq x_2 \leq \frac{26}{5} \wedge \frac{7}{2} \leq x_3 \leq \frac{37}{10}.$$

Once more, using a decision procedure for real arithmetic, we can check that the following sentence is true:

$$\forall\, \boldsymbol{x} \in \mathbb{R}^3.\ \mathrm{BoundBox} \rightarrow V(\boldsymbol{x}) \leq 1400.$$

[8] Intel i5-2520M CPU @ 2.50GHz, 4GB RAM, running Arch Linux kernel 4.2.5-1.



(a) Verified integration up to time $t = 12.7$ from a point initial condition at the origin.

(b) Verified integration up to time $t = 12.2$ from an interval initial condition.

Figure 2: Verified integration using Flow*.

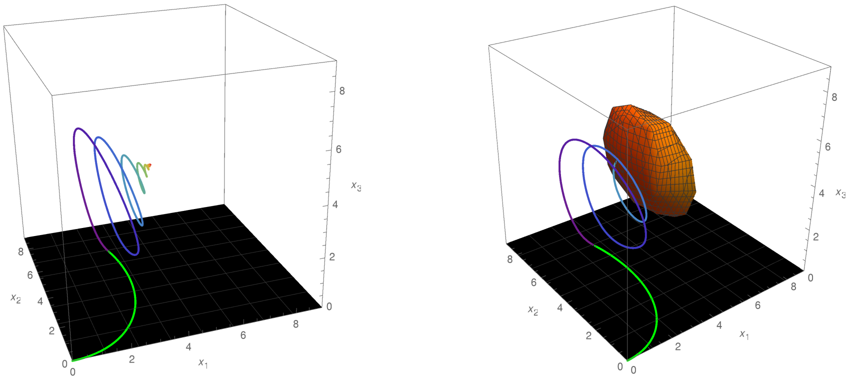
If we are able to establish the following facts:

1. $I \to \Box_{[0,\infty)}\, I$ (I is a continuous invariant),
2. $I \to \text{Steady}$ (inside I, there are no harmful oscillations), and
3. $\text{Init} \to \Diamond_{[0,t]}\, I$ (the system enters the region I in finite time),

then we can conclude that $\text{Init} \to \Diamond_{[0,t]}\, \Box_{[0,\infty)}\, \text{Steady}$ is also true and the system does not exhibit harmful stick-slip oscillations when started inside Init. By taking Init to be the origin $x_1 = 0 \wedge x_2 = 0 \wedge x_3 = 0$, I to be the positively invariant sub-level set $V(\boldsymbol{x}) \leq 1400$ and Steady to be $x_3 > 0$, we are able to conclude the temporal property:

$$x_1 = 0 \wedge x_2 = 0 \wedge x_3 = 0 \to \Diamond_{[0,t]}\, \Box_{[t,\infty)}\, x_3 > 0.$$

Verified integration using Taylor models also allows us to consider *sets* of possible initial conditions, rather than initial points (illustrated in Fig. 2b). This is useful when there is uncertainty about the system's initial configuration; however, in practice this comes with a significant performance overhead for verified integration.



(a) Simulation showing stabilization with positive bit angular velocity.

(b) Simulation showing eventual entry into an ellipsoidal invariant.

Figure 3: Simulation of the hybrid system initialised at the origin with $W_{ob} = 50,000$ N and $u = 6000$ Nm. The trajectory is contained by the flowpipes shown in Fig. 2a and is observed to enter the positively invariant ellipsoid $V(\boldsymbol{x}) \leq 1400$, illustrating the persistence property of eventual absence of stick-slip oscillations.

5 Outlook and Challenges to Automation

Correctness of reachability analysis tools based on verified integration is a soundness critical to the overall verification approach, which makes for a strong case in favour of using formally verified implementations. At present few are available, e.g. see recent work by Immler [20] which presented a formally verified continuous reachability algorithm based on adaptive Runge-Kutta methods. Verified implementations of Taylor model-based reachability analysis algorithms for continuous and hybrid systems would clearly be very valuable. One alternative to over-approximating reachable sets of continuous systems using flowpipes is based on simulating the system using a finite set of sampling trajectories and employs *sensitivity analysis* to address the coverage problem. This technique was explored by Donzé and Maler in [10]. A similar approach employing *matrix measures* has more recently been studied by Maidens and Arcak [28,27].

As an alternative to using verified integration, a number of deductive methods are available for proving eventuality properties in continuous and hybrid systems (e.g. [42,55]). These approaches can be much more powerful since they allow one to work with more general classes of initial and target regions that are necessarily out of scope for methods based on verified integration (e.g. they can work with initial sets that are unbounded, disconnected, etc.) Making effective use of the deductive verification tools currently in existence typically requires significant input and expertise on part of the user (finding the right invariants being one of the major stumbling blocks in practice), in stark contrast to the near-complete level of automation offered by tools based on verified integration. Methods for automatic continuous invariant generation are crucial to the mechanization of the overall verification approach. Progress on this problem would be hugely enabling for non-experts and specialists alike, as it would relieve them from the task of manually constructing appropriate invariants, which often requires intuition and expertise. Work in this area is ongoing (see e.g. [43,25,54]). Indeed, progress on this problem is also crucial to providing a greater level of automation in deductive verification tools.

6 Related Work

Combining elements of qualitative and quantitative reasoning[9] to study the behaviour of dynamical systems has previously been explored in the case of planar systems by Nishida et al. [39]. The idea of combining bounded-time reachability analysis with qualitative analysis in the form of discrete abstraction was investigated by Clarke et al. in [8]. Similar ideas are employed by Carter [6] and Navarro-López in [35], where the concept of *deadness* is introduced and used as a way of disproving liveness properties. Intuitively, deadness is a formalization of an idea that inside certain regions the system cannot be live, i.e. some desired property may never become true as the system evolves inside a "deadness region". These ideas were used in a case study [6, Chapter 5] also featuring the drill system studied in [34], but with a different set of control parameters and in which the verification objective was to prove the existence of a *single trajectory* for which the drill eventually gets "stuck", which is sufficient to disprove the liveness (oscillation) property.

Region stability is similar to our notion of persistence [45], which requires all trajectories to eventually reach some region of the state space. Sound and complete proof rules for establishing region stability have been explored and automated [47], as have more efficient encodings of the proof rule that scale better in dimensionality [31]. However, all algorithms we are aware of for checking region stability require linear or simpler (timed or rectangular) ODEs [45,47,46,31,11,48]. Strong attractors are basins of attraction where every state in the state space eventually reaches a region of the state space [45]. Some algorithms do not check region stability, but actually check stronger properties such as strong attraction, that imply region stability [45]. In contrast to these works, our method checks the weaker notion of persistence for nonlinear ODEs.

She and Ratschan studied methods of proving set eventuality in continuous systems under constraints using Lyapunov-like functions [50]. Duggirala and Mitra also employed Lyapunov-like function concepts to prove inevitability properties in hybrid systems [12]. Möhlmann et al. developed Stabhyil [33], which can be applied to nonlinear hybrid systems and checks classical notions of Lyapunov stability, which is a strictly stronger property than persistence. In [32] Möhlmann et al. extended their work and applied similar ideas, using information about (necessarily invariant) sub-level sets of Lyapunov functions to terminate reachability analysis used for safety verification. Prabhakar and Soto have explored abstractions that enable proving stability properties without having to search for Lyapunov functions, albeit these are not currently applicable to nonlinear systems [48]. In summary, in contrast to other works listed above, our approach enables proving persistence properties in conjunction with safety properties

[9] e.g numerical solution computation with "qualitative" features, such as invariance of certain regions.

for nonlinear, non-polynomial hybrid systems and does not put restrictions on the form or the type of the invariant used in conjunction with bounded time reachability analysis.

7 Conclusion

This paper explored a combined technique for safety and persistence verification employing continuous invariants and reachable set computation based on constructing flowpipes. The approach was illustrated on a model of a simplified oil well drill string system studied by Navarro-López et al., where the verification objective is to prove absence of damaging stick-slip oscillations. The system was useful in highlighting many of the existing practical challenges to applying and automating the proposed verification method. Many competing approaches already exist for verifying safety in hybrid systems, but these rarely combine different methods for reachability analysis and deductive verification, which our approach combines. We demonstrate that a combination of different approaches can be more practically useful than each constituent approach taken in isolation.

Acknowledgements The authors wish to thank the anonymous reviewers for their careful reading and valuable suggestions for improving this paper.

References

1. CAPD library. Online `http://capd.ii.uj.edu.pl/`
2. Akbarpour, B., Paulson, L.C.: MetiTarski: An automatic theorem prover for real-valued special functions. Journal of Automated Reasoning 44(3), 175–205 (2010)
3. Alur, R., Courcoubetis, C., Henzinger, T.A., Ho, P.H.: Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. pp. 209–229 (1992)
4. Berz, M., Makino, K.: Verified integration of ODEs and flows using differential algebraic methods on high-order Taylor models. Reliable Computing 4(4), 361–369 (1998)
5. Blanchini, F.: Set invariance in control. Automatica 35(11), 1747–1767 (1999)
6. Carter, R.A.: Verification of liveness properties on hybrid dynamical systems. Ph.D. thesis, University of Manchester, School of Computer Science (2013)
7. Chen, X., Ábrahám, E., Sankaranarayanan, S.: Flow*: An analyzer for non-linear hybrid systems. In: CAV. pp. 258–263 (2013)
8. Clarke, E.M., Fehnker, A., Han, Z., Krogh, B.H., Ouaknine, J., Stursberg, O., Theobald, M.: Abstraction and counterexample-guided refinement in model checking of hybrid systems. International Journal of Foundations of Computer Science 14(4), 583–604 (2003)
9. Collins, G.E.: Hauptvortrag: Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In: Automata Theory and Formal Languages. pp. 134–183 (1975)
10. Donzé, A., Maler, O.: Systematic simulation using sensitivity analysis. In: HSCC. pp. 174–189 (2007)
11. Duggirala, P.S., Mitra, S.: Abstraction refinement for stability. In: 2011 IEEE/ACM International Conference on Cyber-Physical Systems, ICCPS. Proceedings, pp. 22–31 (Apr 2011)
12. Duggirala, P.S., Mitra, S.: Lyapunov abstractions for inevitability of hybrid systems. In: HSCC. pp. 115–124. ACM, New York, NY, USA (2012)
13. Eggers, A., Ramdani, N., Nedialkov, N.S., Fränzle, M.: Improving the SAT modulo ODE approach to hybrid systems analysis by combining different enclosure methods. Software and System Modeling 14(1), 121–148 (2015)
14. Frehse, G., Le Guernic, C., Donzé, A., Cotton, S., Ray, R., Lebeltel, O., Ripado, R., Girard, A., Dang, T., Maler, O.: SpaceEx: Scalable Verification of Hybrid Systems. In: CAV (2011)
15. Fulton, N., Mitsch, S., Quesel, J.D., Völp, M., Platzer, A.: KeYmaera X: An axiomatic tactical theorem prover for hybrid systems. In: CADE (2015)

16. Ghorbal, K., Platzer, A.: Characterizing algebraic invariants by differential radical invariants. In: TACAS. pp. 279–294 (2014)
17. Ghorbal, K., Sogokon, A., Platzer, A.: A hierarchy of proof rules for checking differential invariance of algebraic sets. In: VMCAI. pp. 431–448 (2015)
18. Gulwani, S., Tiwari, A.: Constraint-based approach for analysis of hybrid systems. In: Gupta, A., Malik, S. (eds.) CAV, LNCS, vol. 5123, pp. 190–203. Springer (2008)
19. Henzinger, T.A.: The theory of hybrid automata. pp. 278–292. IEEE Comp. Soc. Press (1996)
20. Immler, F.: Verified reachability analysis of continuous systems. In: TACAS (2015)
21. Kong, S., Gao, S., Chen, W., Clarke, E.M.: dReach: δ-reachability analysis for hybrid systems. In: TACAS 2015. pp. 200–205 (2015)
22. Koymans, R.: Specifying real-time properties with metric temporal logic. Real-Time Systems 2(4), 255–299 (1990)
23. Lin, Y., Stadtherr, M.A.: Validated solutions of initial value problems for parametric ODEs. Applied Numerical Mathematics 57(10), 1145–1162 (2007)
24. Liu, J., Lv, J., Quan, Z., Zhan, N., Zhao, H., Zhou, C., Zou, L.: A calculus for hybrid CSP. In: Programming Languages and Systems, pp. 1–15 (2010)
25. Liu, J., Zhan, N., Zhao, H.: Computing semi-algebraic invariants for polynomial dynamical systems. In: EMSOFT. pp. 97–106. ACM (2011)
26. Lygeros, J., Johansson, K.H., Simić, S.N., Zhang, J., Sastry, S.S.: Dynamical properties of hybrid automata. IEEE Transactions on Automatic Control 48(1), 2–17 (2003)
27. Maidens, J.N., Arcak, M.: Reachability analysis of nonlinear systems using matrix measures. IEEE Transactions on Automatic Control 60(1), 265–270 (Jan 2015)
28. Maidens, J.N., Arcak, M.: Trajectory-based reachability analysis of switched nonlinear systems using matrix measures. In: CDC. pp. 6358–6364 (Dec 2014)
29. Makino, K., Berz, M.: Cosy infinity version 9. Nuclear Instruments and Methods in Physics Research Section A 558(1), 346–350 (2006)
30. Matringe, N., Moura, A.V., Rebiha, R.: Generating invariants for non-linear hybrid systems by linear algebraic methods. In: SAS. pp. 373–389 (2010)
31. Mitrohin, C., Podelski, A.: Composing stability proofs for hybrid systems. In: Formal Modeling and Analysis of Timed Systems - 9th International Conference, FORMATS. Proceedings, pp. 286–300 (2011)
32. Möhlmann, E., Hagemann, W., Theel, O.E.: Hybrid tools for hybrid systems - proving stability and safety at once. In: FORMATS. pp. 222–239 (2015)
33. Möhlmann, E., Theel, O.: Stabhyli: A tool for automatic stability verification of non-linear hybrid systems. In: HSCC. pp. 107–112. ACM (2013)
34. Navarro-López, E.M., Carter, R.: Hybrid automata: an insight into the discrete abstraction of discontinuous systems. International Journal of Systems Science 42(11), 1883–1898 (2011)
35. Navarro-López, E.M., Carter, R.: Deadness and how to disprove liveness in hybrid dynamical systems. Theor. Comput. Sci. 642(C), 1–23 (Aug 2016)
36. Navarro-López, E.M., Suárez, R.: Practical approach to modelling and controlling stick-slip oscillations in oilwell drillstrings. In: Control Applications, 2004. Proceedings of the 2004 IEEE International Conference on. vol. 2, pp. 1454–1460. IEEE (2004)
37. Nedialkov, N.S.: Interval Tools for ODEs and DAEs. In: SCAN (2006)
38. Neher, M., Jackson, K.R., Nedialkov, N.S.: On Taylor model based integration of ODEs. SIAM Journal on Numerical Analysis 45(1), 236–262 (2007)
39. Nishida, T., Mizutani, K., Kubota, A., Doshita, S.: Automated phase portrait analysis by integrating qualitative and quantitative analysis. In: Proceedings of the 9th National Conference on Artificial Intelligence. pp. 811–816 (1991)
40. Paulson, L.C.: MetiTarski: Past and Future. In: Beringer, L., Felty, A. (eds.) Interactive Theorem Proving, LNCS, vol. 7406, pp. 1–10. Springer Berlin Heidelberg (2012)

41. Platzer, A.: Differential dynamic logic for hybrid systems. J. Autom. Reasoning 41(2), 143–189 (2008)
42. Platzer, A.: Differential-algebraic dynamic logic for differential-algebraic programs. J. Log. Comput. 20(1), 309–352 (2010)
43. Platzer, A., Clarke, E.M.: Computing differential invariants of hybrid systems as fixedpoints. In: CAV. pp. 176–189 (2008)
44. Platzer, A., Quesel, J.D.: KeYmaera: A hybrid theorem prover for hybrid systems. In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR. pp. 171–178 (2008)
45. Podelski, A., Wagner, S.: Model checking of hybrid systems: From reachability towards stability. In: HSCC. Proceedings, pp. 507–521 (2006)
46. Podelski, A., Wagner, S.: Region stability proofs for hybrid systems. In: FORMATS. Proceedings, pp. 320–335 (2007)
47. Podelski, A., Wagner, S.: A sound and complete proof rule for region stability of hybrid systems. In: HSCC. Proceedings, pp. 750–753. Springer (2007)
48. Prabhakar, P., Garcia Soto, M.: Abstraction based model-checking of stability of hybrid systems. In: Computer Aided Verification - 25th International Conference, CAV. Proceedings. pp. 280–295 (2013)
49. Prajna, S., Jadbabaie, A.: Safety verification of hybrid systems using barrier certificates. In: HSCC. pp. 477–492. Springer (2004)
50. Ratschan, S., She, Z.: Providing a basin of attraction to a target region of polynomial systems by computation of Lyapunov-like functions. SIAM J. Control and Optimization 48(7), 4377–4394 (Jul 2010)
51. Richardson, D.: Some undecidable problems involving elementary functions of a real variable. Journal of Symbolic Logic 33(4), 514–520 (12 1968)
52. Sankaranarayanan, S.: Automatic invariant generation for hybrid systems using ideal fixed points. In: HSCC. pp. 221–230 (2010)
53. Sankaranarayanan, S., Sipma, H.B., Manna, Z.: Constructing invariants for hybrid systems. FMSD 32(1), 25–55 (2008)
54. Sogokon, A., Ghorbal, K., Jackson, P.B., Platzer, A.: A method for invariant generation for polynomial continuous systems. In: VMCAI 2016. pp. 268–288 (2016)
55. Sogokon, A., Jackson, P.B.: Direct formal verification of liveness properties in continuous and hybrid dynamical systems. In: FM 2015. pp. 514–531 (2015)
56. Sogokon, A., Jackson, P.B., Johnson, T.T.: Verifying safety and persistence properties of hybrid systems using flowpipes and continuous invariants. Tech. rep., Vanderbilt University (2017)
57. Strzeboński, A.W.: Cylindrical decomposition for systems transcendental in the first variable. J. Symb. Comput. 46(11), 1284–1290 (2011)
58. Taly, A., Tiwari, A.: Deductive verification of continuous dynamical systems. In: Kannan, R., Kumar, K.N. (eds.) FSTTCS. LIPIcs, vol. 4, pp. 383–394. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2009)
59. Tiwari, A.: Generating box invariants. In: Egerstedt, M., Mishra, B. (eds.) HSCC, LNCS, vol. 4981, pp. 658–661. Springer (2008)
60. Wang, S., Zhan, N., Zou, L.: An Improved HHL Prover: An Interactive Theorem Prover for Hybrid Systems. In: ICFEM. pp. 382–399 (2015)
61. Xue, B., Easwaran, A., Cho, N.J., Fränzle, M.: Reach-avoid verification for nonlinear systems based on boundary analysis. IEEE Transactions on Automatic Control (2016)
62. Zhao, H., Yang, M., Zhan, N., Gu, B., Zou, L., Chen, Y.: Formal verification of a descent guidance control program of a lunar lander. In: FM. pp. 733–748 (2014)
63. Zhao, H., Zhan, N., Kapur, D.: Synthesizing switching controllers for hybrid systems by generating invariants. In: Theories of Programming and Formal Methods - Essays Dedicated to Jifeng He on the Occasion of His 70th Birthday. pp. 354–373 (2013)

Large-Scale Linear Systems from Order-Reduction (Benchmark Proposal)

Hoang-Dung Tran, Luan Viet Nguyen, and Taylor T. Johnson[*]

v0.1, 2016-02-18

Abstract

This benchmark suite is composed of nine examples of large-scale linear systems, ranging in dimensionality in the tens to the low thousands. The benchmarks are derived from diverse fields such as civil engineering and robotics, and are based on similar existing test sets for model-order reduction algorithms in control and numerical analysis. Each example is provided in the SpaceEx XML model format as single-mode hybrid automaton and are compatible with the HyST model transformation tool to support analysis in other verification tools. Some preliminary reachability analysis results for some of the smaller examples (on the order of tens of dimensions) are presented using SpaceEx.

Category: academic **Difficulty:** low through challenge

1 Context and Origins

Symbolic state-space analysis has shown advantages in safety verification of continuous and hybrid systems in which the essential task is computing the set of reachable states symbolically with an iterative algorithm [1]. The main challenge of this approach is state-space explosion, which roughly is that the complexity of computation grows exponentially with the system dimensionality [2]. To implement efficiently symbolic reachability algorithms, significant effort has been invested in finding appropriate representations for the set of states that supports efficient operations used in the iterative computation. From classical polyhedral representations which are used in hybrid systems model checkers such as HyTech [3,4] and **d/dt** [5], more efficient representations such as zonotopes [6–8] and support functions [9,10] have been proposed and integrated in tools such as CORA and SpaceEx that use these state-of-the-art representations for analysis of hybrid systems with linear dynamics.

[*] Computer Science and Engineering, University of Texas at Arlington — Arlington, TX

No.	Benchmark	Type	n	m	p
1	Motor control system (MCS)	LTI	8	2	2
2	Building model (BM) [12]	LTI	48	1	1
3	International space station (ISS) [12]	LTI	270	3	3
4	Partial differential equation (Pde) [12]	LTI	84	1	1
5	FOM [12]	LTI	1006	1	1
6	Modified nodal analysis model 1 [12](MNA-1)	LTI	578	9	9
7	Modified nodal analysis model 5 [12](MNA-5)	LTI	10913	9	9
8	Heat equation [12]	LTI	200	1	1
9	Clamped beam model [12]	LTI	348	1	1

Table 2.1: Benchmarks for the order-reduction abstraction method in which n is dimension of the system; m and p are the number of inputs and outputs respectively.

In spite of these advances, reachability analysis of large-scale systems with hundreds to thousands of dimensions is still infeasible even for linear time invariant (LTI) systems, i.e., without any discrete switching behavior. It is important to develop new techniques and tools that can be used to verify the safety of such high-dimensional systems, which usually exist in a broad range of fields and applications such as control systems, biological systems, analog circuits, and multi-agent systems.

To help test and evaluate reachability analysis methods and tools to enable verification of high-dimensional systems, we construct a set of benchmarks that are essentially LTI systems arising from model order reduction [11, 12]. These benchmarks, which are models of practical systems in different fields, have dimensions varying from ten to thousands. Each benchmark is given in the SpaceEx format as a single-mode hybrid automaton and can be easily transformed to other formats such as dReach [13] or Flow* [14] using the HyST model transformation tool [15]. Reachability analysis of some of the small and medium-size benchmarks (i.e., < 50 dimensions) are presented. These benchmarks may be effective to test and evaluate the scalability of verification approaches when dealing with large-size benchmarks (i.e. > 50 dimensions).

2 Brief descriptions

Since most of benchmarks are high-dimensional, their dynamic equations cannot be presented in detail in this paper. We refer readers to [11, 12] for for further details and derivations, as well as our provided supplementary material.[1] The

[1] The benchmarks are available online, http://verivital.com/hyst/benchmark-large-scale/

Benchmark	**Initial set of states**	**Input constraint**	**Safety specification**
	$X_0 = \{x_0 \in \mathbb{R}^n \vert\ lb(i) \leq x_0(i) \leq ub(i),\ 1 \leq i \leq n\}$	$u = [u_1, \cdots, u_m]^T$	$y = [y_1, \cdots, y_p]^T$
Motor control system	$lb(i) = ub(i) = 0,\ i = 2, 3, 4, 6, 7, 8,$ $lb(2) = 0.002,\ ub(2) = 0.0025,$ $lb(3) = 0.001,\ ub(3) = 0.0015.$	$u_1 \in [0.16,\ 0.3],$ $u_2 \in [0.2,\ 0.4].$	unsafe region: $0.35 \leq y_1 \leq 0.4,$ $0.45 \leq y_2 \leq 0.6.$
Building model	$lb(i) = 0.0002,\ ub(i) = 0.00025,\ 1 \leq i \leq 10,$ $lb(25) = -0.0001,\ ub(25) = 0.0001,$ $lb(i) = ub(i) = 0,\ 11 \leq i \leq 48,\ i \neq 25.$	$u_1 \in [0.8,\ 1].$	unsafe region: $0.006 < y_1$
Partial differential equation	$lb(i) = 0,\ ub(i) = 0,\ 1 \leq i \leq 64$ $lb(i) = 0.001,\ ub(i) = 0.0015,\ 64 \leq i \leq 80,$ $lb(i) = -0.002,\ ub(i) = -0.0015,\ 81 \leq i \leq 84.$	$u_1 \in [0.5,\ 1].$	safe region: $y_1 \leq 12$
International space station	$lb(i) = -0.0001,\ ub(i) = 0.0001,\ 1 \leq i \leq 270.$	$u_1 \in [0,\ 0.1],$ $u_2 \in [0.8,\ 1],$ $u_3 \in [0.9,\ 1].$	Safe region: $-0.0005 \leq y_3 \leq 0.0005$
FOM	$lb(i) = -0.0001,\ ub(i) = 0.0001,\ 1 \leq i \leq 400$ $lb(i) = 0.0002,\ ub(i) = 0.00025,\ 401 \leq i \leq 800,$ $lb(i) = 0,\ ub(i) = 0,\ 801 \leq i \leq 1006.$	$u_1 \in [-1,\ 1].$	safe region: $y_1 \leq 45$
MNA-1	$lb(i) = 0.001,\ ub(i) = 0.0015,\ 1 \leq i \leq 2$ $lb(i) = 0,\ ub(i) = 0,\ 3 \leq i \leq 578,$	$u_i = 0.1,\ 1 \leq i \leq 5,$ $u_i = 0.2,\ 6 \leq i \leq 9.$	unsafe region: $y_1 > 0.5$
MNA-5	$lb(i) = 0.0002,\ ub(i) = 0.00025,\ 1 \leq i \leq 10$ $lb(i) = 0,\ ub(i) = 0,\ 11 \leq i \leq 10913,$	$u_i = 0.1,\ 1 \leq i \leq 5,$ $u_i = 0.2,\ 6 \leq i \leq 9.$	safe region: $y_1 \leq 0.2,\ y_1 \leq 0.15$
Heat equation	$lb(i) = 0.6,\ ub(i) = 0.625,\ 1 \leq i \leq 2$ $lb(i) = 0,\ ub(i) = 0,\ 3 \leq i \leq 200,$	$u_1 \in [-0.5,\ 0.5].$	safe region: $y_1 \leq 0.1$
Clamped beam model	$lb(i) = 0,\ ub(i) = 0,\ 1 \leq i \leq 300$ $lb(i) = 0.0015,\ ub(i) = 0.002,\ 301 \leq i \leq 348,$	$u_1 \in [0.2,\ 0.8].$	unsafe region: $y_1 > 1000$

Table 2.2: Initial states, input constraints and safety specification for the outputs of the benchmarks.

general form of the dynamics is:

$$\dot{x}(t) = Ax(t) + Bu(t)$$
$$y(t) = Cx(t),$$

where $x(t) \in \mathbb{R}^n$ is the *system state*, $y(t) \in \mathbb{R}^p$ is the *system output*, $u(t)$ is the *control input*, $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, and $C \in \mathbb{R}^{p \times n}$.

In this section, we introduce briefly these benchmarks. Table 2.1 summarizes names, number of dimensions, and numbers of inputs and outputs of the benchmarks. The initial set of states, input constraints, and safety specifications of the benchmarks are given in Table 2.2.

Motor control system. The motor control system benchmark includes two motors that are controlled synchronously. Each motor has a local controller that is designed using pole placement method [16] to control the motor to satisfy: 1) the overshoot of the motor position is less than 16%; 2) setting time is less than 0.04 seconds; 3) No steady-state error, even in the presence of a step disturbance input.

Building model. The building model is a model of the Los Angeles University Hospital with 8 floors, each of which has 3 degrees of freedom [11]. This system has 48 state variables in which we are mostly interested in the twenty-fifth state $x_{25}(t)$, which is the motion of the first coordinate. The twenty-fifth state is the interested output of the building model and should not reach to the unsafe region given in Table 2.2.

Partial differential equation. The partial differential equation (PDE) is given by

$$\frac{\partial x}{\partial t} = \frac{\partial^2 x}{\partial z^2} + \frac{\partial^2 x}{\partial v^2} + 20\frac{\partial x}{\partial v} - 180x + f(v,z)u(t),$$

where x is a function of time t, vertical position v and horizontal position z. This problem lies on a square domain defined by two opposite points $(0,0)$ and $(1,1)$. The function $x(t,v,z)$ is zero on the boundaries of the square. A state-space equation of dimension of $N = n_v n_z$ of this PDE can be given by discretizing with centered difference approximation on a grid of $n_v \times n_z$ points. The input vector corresponding to $f(v,z)$ is composed of random elements while the output vector of the system is equated to the input vector for simplicity. The state-space model of PDE covered in this paper corresponds to the case of $n_v = 7$ and $n_z = 12$.

International Space Station (ISS). The ISS state-space model presented in this paper is a structural model of component 1R (Russian service module) of the International Space Station. It has 270 state variables with three inputs and three outputs.

FOM. This is state-space model of a dynamical system with following matrices:

$$A = \begin{bmatrix} A_1 & & & \\ & A_2 & & \\ & & A_3 & \\ & & & A_4 \end{bmatrix},\ A_1 = \begin{bmatrix} -1 & 100 \\ -100 & -1 \end{bmatrix},\ A_2 = \begin{bmatrix} -1 & 200 \\ -200 & -1 \end{bmatrix},$$

$$A_3 = \begin{bmatrix} -1 & 400 \\ -400 & -1 \end{bmatrix},\ A_4 = \begin{bmatrix} -1 & & & \\ & -2 & & \\ & & \ddots & \\ & & & -1000 \end{bmatrix},$$

$$B^T = C = [\underbrace{10 \cdots 10}_{6}\ \underbrace{1 \cdots 1}_{1000}].$$

Modified nodal analysis model. The following Modified Nodal Analysis (MNA) equation is constituted from connecting voltage sources to the ports of a multiport:

$$E\dot{x}_n A = Ax_n + Bu_p,$$
$$i_p = Cx_n,$$

in which i_p and u_p are the port currents and voltages vectors respectively and

$$A = \begin{bmatrix} -N & -G \\ G^T & 0 \end{bmatrix},\ E = \begin{bmatrix} L & 0 \\ 0 & H \end{bmatrix},\ x_n = \begin{bmatrix} v \\ i \end{bmatrix},$$

where v and i are variables of the MNA including node votages, inductor and voltage source currents, respectively. The matrices $-A$ and E represent the conductance and susceptance matrices. The matrices $-N$, L and H contains the stamps for resistors, capacitors and inductors, respectively. Matrix G consists of 1, -1 and 0, which describe the current variables in Kirchhoff's Current Law (KCL) equation. The input matrix B and output matrix C satisfy $B = C^T$. We give two MNA models with different number of state variables in the paper.

Heat equation. The state-space model of Heat equation is giving by discretizing the following equation:

$$\left\{\begin{array}{ll} PDE & \frac{\partial}{\partial t}T(x,t) = \alpha\frac{\partial^2}{\partial x^2}T(x,t) + u(x,t),\ x \in (0,1);\ t > 0, \\ BCs & T(0,t) = 0 = T(1,t),\ t > 0, \\ IC & T(x,0) = 0,\ x \in (0,1). \end{array}\right\}$$

where $T(x,t)$ represents the temperature field on a thin rod and $u(x,t)$ is the heat source.

Clamped beam model. The state-space clamped beam model, which is obtained by spatial discretization of an appropriate partial different equation, has 348 states, one input and one output in which the input represents the force applied to the structure and the output is the displacement.

3 Reachability analysis

Since all benchmarks are LTI systems, there are different tools that can be used to analyze the safety of these benchmarks such as SpaceEx [10], CORA [17], CheckMake [18], DReach [13], and Flow* [14]. We specify each benchmark in the SpaceEx format as a single-mode hybrid automaton, which can be easily transformed to other formats using HyST [15].

Table 3.1 presents a preliminary overview of the computation cost of time-bounded reachability analysis for the benchmarks using SpaceEx. These experiments are conducted on a personal computer with the following configurations: Intel (R) Core(TM) i7-2677M CPU at 1.80GHz, 4GB RAM, and 64-bit Window 7. The reachability analysis is conducted in a bounded time range $[0, 20s]$.

Benchmark	**LGG**	**STC**
	Time(s)	**Time(s)**
Motor control system	27	N/A
Building model	893	N/A
Partial differential equation	OOT	N/A
International space station	OOT	N/A
FOM	OOT	N/A
MNA-1	OOT	N/A
MNA-5	OOT	N/A
Heat equation	OOT	N/A
Clamped beam model	OOT	N/A

Table 3.1: Computation cost for verification of the benchmarks using SpaceEx [10] with two scenarios LGG [19] and STC [20]. The terms of "N/A" and "OOT" mean "not applicable" and "out of time".

The SpaceEx scenarios tested are LGG [19] and STC [20]. The sampling time is selected as 0.001 for all benchmarks. We note that the sampling time and time horizon should be selected appropriately based on the dynamics of specific system, for example, using the rule of thumb to pick the sampling time based on the inverse of the maximum eigenvalue. Intuitively, this would mean to pick large sampling times for slow dynamics and small sampling times for fast dynamics. Thus, while our preliminary results as shown in Table 3.1 indicate some examples are infeasible for analysis with SpaceEx, it is possible that a more careful selection of parameters would enable analysis of these systems, and we hope other researchers will be interested to try these examples. We set the upper limit for SpaceEx running time as two hours, and an experiment is said to be out of time (OOT) if we can not get the result after two hours. The reason the STC scenario did not produce results is due to the use of outputs as invariant conditions (i.e., $y = Cx$) with nondeterministic dynamics, which does not seem to be supported when using STC.

Next, we present briefly the reachability analysis of some small and medium-size benchmarks (i.e., less than 50 dimensions).

Motor control system. Figure 3.1 depicts the reachable set of the interested states of the motor control system. As shown in the figure, the reachable set does not reach to the unsafe region. Thus, we can conclude that the system is safe in the bounded time $[0, 20s]$. A stronger conclusion about the safety of the motor control system may be given by considering unbounded time safety verification.

Building model. Figure 3.2 depicts the reachable set of the this state of the building model. As can be seen from the figure, the reachable states of the output do not intersect the unsafe region. Thus, we can conclude that the system is safe in the bounded time $[0, 20s]$. Similar to the above motor control

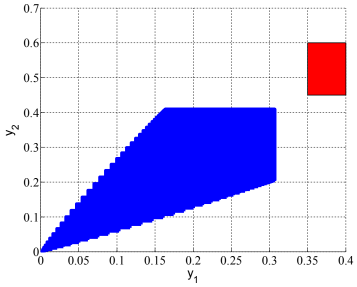


Figure 3.1: Reachable set of interested outputs of the motor control system (in $[0, 20s]$) and its corresponding unsafe region (the red region). The reachable set of interested outputs do not reach the unsafe region, thus the system is safe (in a bounded time interval $[0, 20s]$).

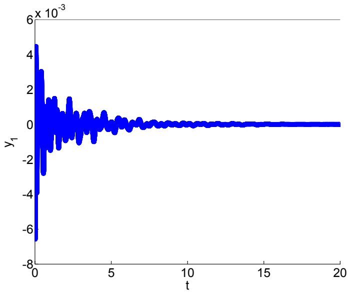


Figure 3.2: Reachable set of interested output of the building model system (in $[0, 20s]$) and its corresponding unsafe region (the region above the red line). The reachable set of interested output do not reach the unsafe region, thus the system is safe (in a bounded time interval $[0, 20s]$).

system, a stronger conclusion about the safety of the building model may be given by considering unbounded time safety verification.

4 Outlook

Overall, we present in this paper a set benchmarks for purely continuous linear systems (i.e., LTI systems), modeled as single-mode hybrid automata in the SpaceEx model format. The benchmarks range in dimensionality from tens to thousands of dimensions, and come from many different domains. The continuous and hybrid verification community may use these benchmarks for comparing methods and tools, especially with respect to continuous post operator benchmarking for systems with a high number of dimensions. In ongoing and future work, we intend to introduce additional high-dimensional benchmarks with both piecewise affine dynamics and continuous dynamics including ones originally encoded as differential algebraic equations (DAEs), and are also investigating formalization of order-reduction methods as sound abstractions using approximate bisimulation relations [21].

Acknowledgements The material presented in this paper is based upon work supported by the National Science Foundation (NSF) under grant numbers CNS 1464311 and CCF 1527398, the Air Force Research Laboratory (AFRL) through contract number FA8750-15-1-0105, and the Air Force Office of Scientific Research (AFOSR) under contract number FA9550-15-1-0258. The U.S. government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of AFRL, AFOSR, or NSF.

References

[1] P. Tabuada, *Verification and control of hybrid systems: a symbolic approach.* Springer Science & Business Media, 2009.

[2] R. Pelánek, "Fighting state space explosion: Review and evaluation," in *Formal Methods for Industrial Critical Systems.* Springer, 2008, pp. 37–52.

[3] R. Alur, T. A. Henzinger, and P.-H. Ho, "Automatic symbolic verification of embedded systems," *Software Engineering, IEEE Transactions on*, vol. 22, no. 3, pp. 181–201, 1996.

[4] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi, "Hytech: A model checker for hybrid systems," in *Computer aided verification.* Springer, 1997, pp. 460–463.

[5] E. Asarin, O. Bournez, T. Dang, and O. Maler, "Approximate reachability analysis of piecewise-linear dynamical systems," in *Hybrid Systems: Computation and Control.* Springer, 2000, pp. 20–31.

[6] A. Girard, "Reachability of uncertain linear systems using zonotopes," in *Hybrid Systems: Computation and Control.* Springer, 2005, pp. 291–305.

[7] A. Girard, C. Le Guernic, and O. Maler, "Efficient computation of reachable sets of linear time-invariant systems with inputs," in *Hybrid Systems: Computation and Control.* Springer, 2006, pp. 257–271.

[8] M. Althoff, O. Stursberg, and M. Buss, "Computing reachable sets of hybrid systems using a combination of zonotopes and polytopes," *Nonlinear Analysis: Hybrid Systems*, vol. 4, no. 2, pp. 233–249, 2010.

[9] C. Le Guernic and A. Girard, "Reachability analysis of hybrid systems using support functions," in *Computer Aided Verification.* Springer, 2009, pp. 540–554.

[10] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, "Spaceex: Scalable verification of hybrid systems," in *Computer Aided Verification.* Springer, 2011, pp. 379–395.

[11] A. C. Antoulas, D. C. Sorensen, and S. Gugercin, "A survey of model reduction methods for large-scale systems," *Contemporary Mathematics*, vol. 280, pp. 193–219, 2001.

[12] Y. Chahlaoui and P. Van Dooren, "A collection of benchmark examples for model reduction of linear time invariant dynamical systems." 2002.

[13] S. Kong, S. Gao, W. Chen, and E. Clarke, "dreach: δ-reachability analysis for hybrid systems," pp. 200–205, 2015.

[14] X. Chen, E. Ábrahám, and S. Sankaranarayanan, "Flow*: An analyzer for non-linear hybrid systems," in *Computer Aided Verification.* Springer, 2013, pp. 258–263.

[15] S. Bak, S. Bogomolov, and T. T. Johnson, "Hyst: a source transformation and translation tool for hybrid automaton models," in *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control.* ACM, 2015, pp. 128–133.

[16] R. C. Dorf and R. H. Bishop, "Modern control systems," 1998.

[17] M. Althoff, "An introduction to cora 2015," in *Proc. of the Workshop on Applied Verification for Continuous and Hybrid Systems*, 2015.

[18] B. I. Silva, K. Richeson, B. Krogh, and A. Chutinan, "Modeling and verifying hybrid dynamic systems using checkmate," in *Proceedings of 4th International Conference on Automation of Mixed Processes*, vol. 4. Citeseer, 2000, pp. 1–7.

[19] C. Le Guernic and A. Girard, "Reachability analysis of linear systems using support functions," *Nonlinear Analysis: Hybrid Systems*, vol. 4, no. 2, pp. 250–262, 2010.

[20] G. Frehse, R. Kateja, and C. Le Guernic, "Flowpipe approximation and clustering in space-time," in *Proceedings of the 16th international conference on Hybrid systems: computation and control.* ACM, 2013, pp. 203–212.

[21] H.-D. Tran, L. V. Nguyen, W. Xiang, and T. T. Johnson, "Order-reduction abstractions for safety verification of high-dimensional linear systems," *arXiv*, 2016.

EPiC Series in Computing

Volume XXX, 2017, Pages 1–12

ARCH17. 4th International Workshop on Applied Verification of Continuous and Hybrid Systems

Distributed Autonomous Systems (Benchmark Proposal)

Hoang-Dung Tran, Luan Viet Nguyen, Weiming Xiang and Taylor T. Johnson

Vanderbilt University, Tennessee, USA
taylor.johnson@gmail.com

Abstract

This benchmark suite consists of a number of examples of autonomous multi-agent systems where the agent number ranges from two to ten. The benchmarks are derived from the field of position-based formation control in autonomous robotics and vehicles. Their models are given as network of hybrid automata in the SpaceEx XML model format and can be transformed to other verification tools model formats using HyST, a model transformation tool. Safety of a small benchmark with two agents is analyzed using SpaceEx.
Category: academic **Difficulty:** low through challenge

1 Context and Origins

Intelligent autonomous systems have been a “hot” research topic for many years because of its rigorous application domains such as robotics, unmanned aerial vehicles (UAV), autonomous cars and sensors networks. The challenges in modeling, analysis, design and testing a such intelligent system have attracted researchers from different disciplines such as biology, computer, communication and control. In an early step, the intelligent behavior called “flocking behavior” of a group of animals such as bird, insect and fish has been investigated deeply over decades in the field of biology [1]. The behavior has been first modeled and simulated using computer in [2]. This work has inspired a new field of modeling, control and design for autonomous systems which is now considerably an important topic for the next generation of modern technology.

Consensus and formation controls are two fundamental problems in designing an autonomous system that perform an intelligent behavior. Control scientists have proposed numerous protocols over last decades to drive the system to achieve some control objectives [3–9]. Generally, to perform a specific task, the agents need to exchange their information and cooperate with each other over communication channel. The communication topology of an autonomous system describes in detail how the information *flow* in the system. The communication topology can be *static*, i.e. does not change over times, or *dynamics*, i.e. may change over times. It can also be *directed*, i.e. information *flows* in one direction over a connection between two agents, or *undirected*, i.e. the information *flows* in both directions over a connection between two agents. The communication topology expresses the sensing and communicating capacities of the agents which affect significantly to the stability, controllability and the convergence of an autonomous system. Graph theory has been proved as an powerful tool to model the communication topology and analyze the controllability of autonomous systems [10].

Formation control for autonomous systems [7–9] is seeking control laws to guarantee that the agents move to pre-determined positions while keeping the system formation in some specific shapes when moving. Depending on the sensing and communicating capacities of the agents, i.e. the communication topology, the formation control strategies can be categorized into position-based, displacement-based and distance-based approaches [11]. One essential safety requirement for the system is that there is no collision when the agents are moving. These formation control strategies have shown *informally* the ability of the agents avoiding collision via simulation-based testing. To guarantee the safety of the system, its formal model need to be given and verified using formal verification techniques.

Toward safety and liveness requirements of autonomous systems, some control algorithms have been proposed and verified using formal verification techniques recently [12, 13]. In this context, the formal model of an autonomous system is given based on discrete time intervals and to guarantee the safety of the system, the controller usually can perform some particular actions to resolve the potential risks coming. The whole system is modeled as a labeled transition system and the safety and liveness requirements are written in form of linear temporal logic (LTL).

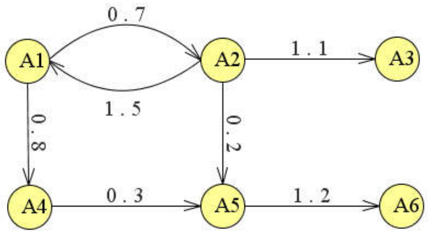
Inspired by above interesting works, in this paper, we obtain a set of autonomous systems benchmarks written in SpaceEx XML format. Each agent is modeled separately as a single hybrid automaton and the whole system is a network of hybrid automata which is basically a composition of all agents. Different from [12,13], these benchmarks have *continuous dynamics*. Therefore, their safety requirements can be verified using existing verification tools that support verifying continuous dynamics [14–17]. In addition, when the number of agents increases, the benchmark models become larger that makes them harder to be verified. Thus, our benchmark suite is also useful for testing the scalability of verification tools.

The rest of the paper is organized as follows: Section 2 presents the description of an autonomous system including the communication topology, the motion dynamics of the agents and the position-based formation control strategies. Section 3 gives the safety analysis of some small autonomous systems using SpaceEx. Section 4 discusses some interesting issues for the future work and concludes the paper.

2 System descriptions

2.1 Communication topology

Directed/undirected graphs are powerful tool for modeling the interaction between agents in an autonomous system. In this benchmark suite, the communication topologies of all autonomous systems are modeled using directed graphs. A digraph (directed graph) defined by a tuple $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a finite non-empty set of vertices and $\mathcal{E} \in \mathcal{V}^2$ is a set of *ordered* pairs of vertices, called edges. It can be understood that vertice $v_i \in \mathcal{V}$ represents for the i^{th} agent an autonomous system and ordered edge (i, j) represents for the interaction between the agent i and the agent j where the information flows from i to j, i.e. agent j receives the information from agent i. To model how much information flows in communication, we use a weighted digraph which can be defined by an adjacency matrix $A = [a_{ij}]_{n \times n}$, where $a_{ii} = 0$, $a_{ij} > 0$ if $(j, i) \in \mathcal{E}$ and $n = |\mathcal{V}|$ is the number of agents in the system. Figure 2.1 illustrates an example of communication topology of an autonomous system with six agents [18]. From the communication topology, it can be seen that one agent only can collect some information from its neighbors, not from all other agents.



$$A = \begin{bmatrix} 0 & 1.5 & 0 & 0 & 0 & 0 \\ 0.7 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.1 & 0 & 0 & 0 & 0 \\ 0.8 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.2 & 0 & 0.3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.2 & 0 \end{bmatrix}$$

Figure 2.1: An example of communication topology using a weighted digraph.

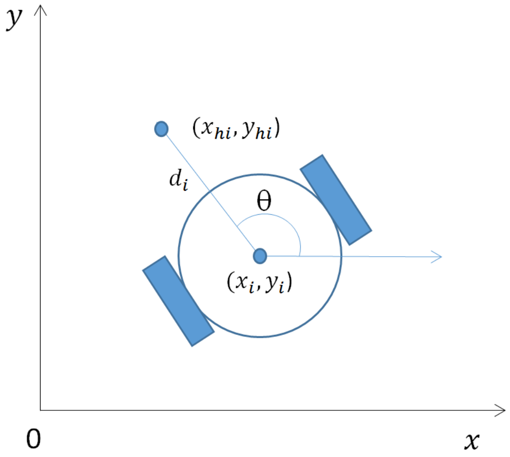


Figure 2.2: Non-holonomic differential driven mobile robot.

A communication topology can be static, as in the case of the example, or dynamic, i.e. the connections between agents can be varied over times. A dynamic communication topology may be convenient to characterize naturally the interaction behaviors of agents in practice where the sensing capacity of agents is limited in some ranges and hence, it can not recognize the other agents outside of its sensing range. However, the dynamic communication topology increases the difficulty in designing the control law to guarantee autonomous systems to perform the intelligent flocking behavior. In this paper, the benchmarks can be categorized into static or dynamic communication topology.

We have briefly introduced modeling interaction between agents in an autonomous system using directed graph. Next, we give the dynamics of the agents and the formation control rules of the autonomous system.

2.2 Motion dynamic and formation control

In this paper, we consider the formation control for multiple mobile robots in a 2-dimensional plan in which the equations of motion of a non-holonomic mobile robot depicted in Figure 2.2 are given by

$$\begin{aligned} \dot{x}_i &= v_i \cos(\theta_i), \\ \dot{y}_i &= v_i \sin(\theta_i), \\ \dot{\theta}_i &= \omega_i, \\ m_i \dot{v}_i &= f_i, \\ J_i \dot{\omega}_i &= \tau_i, \end{aligned} \tag{2.1}$$

where (x_i, y_i) is the Cartesian position of the robot centre, θ_i is the orientation, v_i is the linear velocity, ω_i is the angular velocity, m_i is the mass, J_i is the mass moment of inertia, f_i is the force, and τ_i is the torque applied to the robot.

Since Equation 2.1 contains the nonlinear functions $\cos(\theta_i)$ and $\sin(\theta_i)$, the robot dynamic is nonlinear and thus, we cannot model and analyze the system using SpaceEx. Fortunately, we can avoid the non-holonomic constraint and obtain a linear model for the system by introducing intermediate position variables (x_{hi}, y_{hi}) as follows [18].

$$\begin{bmatrix} x_{hi} \\ y_{hi} \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} + d_i \begin{bmatrix} \cos(\theta_i) \\ \sin(\theta_i) \end{bmatrix} \tag{2.2}$$

We can see that (x_{hi}, y_{hi}) is a position off the wheel axis of the i^{th} robot by a distance d_i. Now, if we let

$$\begin{bmatrix} f_i \\ \tau_i \end{bmatrix} = \begin{bmatrix} \frac{1}{m_i}\cos(\theta_i) & -\frac{d_i}{J_i}\sin(\theta_i) \\ \frac{1}{m_i}\sin(\theta_i) & -\frac{d_i}{J_i}\cos(\theta_i) \end{bmatrix}^{-1} \begin{bmatrix} v_{xi} + v_i\omega_i \sin(\theta_i) + d_i\omega_i^2\cos(\theta_i) \\ v_{yi} - v_i\omega_i\cos(\theta_i) + d_i\omega_i^2\sin(\theta_i) \end{bmatrix}$$

Then we can obtain the new linear equations of motion for each robot as a double-integrator system:

$$\begin{aligned} \dot{x}_{hi} &= v_{xi}, \\ \dot{v}_{xi} &= b_{xi}, \\ \dot{y}_{hi} &= v_{yi}, \\ \dot{v}_{yi} &= b_{yi}. \end{aligned} \tag{2.3}$$

The control objective is to drive the mobile robots from their initial location (x_{hi}^0, y_{hi}^0) to predefined destinations (x_{hi}^d, y_{hi}^d) while preserving the formation of the system during the transition, e.g., a square formation for 4-robots team, a triangle formation for 3-robots team. Assume we have the communication topology of the system defined by adjacent matrix $A = [a_{ij}]_{n \times n}$, where n is the number of agents in the system, the *position-based* formation control law for the system is designed as follows [18].

$$\begin{aligned} b_{xi} &= -\alpha_x(x_{hi} - x_{hi}^d) - \gamma_x\alpha_x\dot{x}_{hi} - \sum_{j=1}^{n} a_{ij}[(x_{hi} - x_{hi}^d) - (x_{hj} - x_{hj}^d)] - \sum_{j=1}^{n} \gamma_x a_{ij}(\dot{x}_{hi} - \dot{x}_{hj}) \\ b_{yi} &= -\alpha_y(y_{hi} - y_{hi}^d) - \gamma_y\alpha_y\dot{y}_{hi} - \sum_{j=1}^{n} a_{ij}[(y_{hi} - y_{hi}^d) - (y_{hj} - y_{hj}^d)] - \sum_{j=1}^{n} \gamma_y a_{ij}(\dot{y}_{hi} - \dot{y}_{hj}) \end{aligned} \tag{2.4}$$

where $\alpha_* > 0$ and $\gamma_* > 0$.

The first two terms of the control law are responsible for driving each robot to its destination (goal seeking) while the last two terms of the control law are to preserve the formation between robots (formation keeping). In term of verification, there are both liveness and safety properties need to be verified. The liveness property relates to goal seeking objective as we need to guarantee that each robot *finally reach its destination*. The safety property concerns the formation keeping problem as it is required there is *no collision* when robots are moving.

With above formation control law, we can derive the *closed-loop* dynamic equation for the system. Let $x_{ei} = x_{hi} - x_{hi}^d$, $y_{ei} = y_{hi} - y_{hi}^d$, $x_e = [x_{e1}, ..., x_{en}]^T$ and $y_e = [y_{e1}, ..., y_{en}]^T$, the closed-loop dynamic of the system can be written by

$$\begin{aligned} \begin{bmatrix} \dot{x}_e \\ \ddot{x}_e \end{bmatrix} &= \begin{bmatrix} 0_{n\times n} & I_n \\ -(L + \alpha_x I_n) & -\gamma_x(L + \alpha_x I_n) \end{bmatrix} \begin{bmatrix} x_e \\ \dot{x}_e \end{bmatrix} \\ \begin{bmatrix} \dot{y}_e \\ \ddot{y}_e \end{bmatrix} &= \begin{bmatrix} 0_{n\times n} & I_n \\ -(L + \alpha_y I_n) & -\gamma_y(L + \alpha_y I_n) \end{bmatrix} \begin{bmatrix} y_e \\ \dot{y}_e \end{bmatrix} \end{aligned} \tag{2.5}$$

where $0_{n\times n}$ is n-dimensional square zero matrix, I_n is n-dimensional identity matrix and $L = [l_{ij}]_{n\times n}$ in which $l_{ii} = \sum_{j\neq i} a_{ij}$ and $l_{ij} = -a_{ij}$, where $i \neq j$.

We have already described the communication topology, the system dynamics and formation control law. Next, we formally define the safety property for the system.

3 Safety property

Informally, the system is safe if there is no collision when the robots move to their destination. In other word, the distance between two arbitrary robots (i.e., the distance between their centers) need to be larger than the diameter of the robots. Recall that the robots shapes are circles and their sizes are identical. The distance between the i^{th} and j^{th} robots is

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}.$$

Let $\mathcal{D}$ be the diameter of the robot. The safety property S of the system can be defined formally as follows

$$S: \ \forall i, j, i \neq j, t \geq 0, d_{ij} > \mathcal{D}. \tag{3.1}$$

The dual unsafe specification U for two arbitrary robots can be defined by the following circle.

$$U : (x_i - x_j)^2 + (y_i - y_j)^2 \leq \mathcal{D}^2. \tag{3.2}$$

From Equation 2.2, we have:

$$\begin{aligned} (x_{hi} - x_{hj}) - (d_i + d_j) \leq (x_i - x_j) \leq (x_{hi} - x_{hj}) + (d_i + d_j) \\ (y_{hi} - y_{hj}) - (d_i + d_j) \leq (y_i - y_j) \leq (y_{hi} - y_{hj}) + (d_i + d_j) \end{aligned} \tag{3.3}$$

The above inequality shows that we can compute the reachable sets of $(x_i - x_j)$ and $(y_i - y_j)$ by bloating the reachable sets of $(x_{hi} - x_{hj})$ and $(y_{hi} - y_{hj})$ by $(d_i + d_j)$. Then, using the bloated reachable sets, we can check whether they violate the safety property (i.e., whether the reachable sets reach the corresponding unsafe region defined in Equation 3.2).

We have formally defined the safety property of the system and described briefly how to check the safety of the system. Next, we discuss how to model the distributed autonomous system using hybrid automata.

4 System modeling

There are three approaches for modeling an autonomous system using hybrid automata framework. The first approach is that we can model the system using *decentralized style* in which each agent as a hybrid automata network composed by *dynamic component* describing the dynamic of the agent as defined in Equation 2.3 and *controller component* describing the distributed formation control law in Equation 2.4. Since the communication topology of the autonomous system may change, the controller component may switch its operation between different modes. The whole system will be a network of hybrid automata composing n agent's models. In the second approach, we can model the system using *centralized style* in which each agent is a single-mode hybrid automaton describing the dynamic of the agent, the control law given in Equation 2.4 is modeled as a *centralized coordinator* which is a hybrid automata containing one or multiple modes. Last but not least, we can also model the system as one single automaton describing the closed-loop dynamic defined by Equation 2.5.

The first two modeling approaches have two advantages. First, they describe intuitively the hierarchical architecture of the system in which each agent is a separate entity. The obtained model in the first modeling approach illustrates the *decentralized control strategy* in autonomous systems where the control signal is computed at agent side. In contrast to decentralized control strategy, the second approach describes the *centralized control strategy* where the coordinator collects the information of all agents and computes the control signals before sending them to the agents. Second, since the first two modeling approaches separate the agent's dynamic and the control law, they are convenient for changing the dynamics of the agents and they also allow modeling the switching happen between different dynamics of one agent. In addition, it is easy to model and verify the system under a complex hybrid control law when the controller switches between different modes along with communication topology changes. While the first two modeling approaches are convenient for modeling complex autonomous systems, the third approach is useful for finding an abstraction for the whole system that allows us to verify a very large autonomous system using order-reduction abstraction method [19]. In this benchmark suite, we use the first and the second approaches to model distributed autonomous systems. Examples of these modeling approaches are depicted in Figure 4.1 and Figure 4.2.

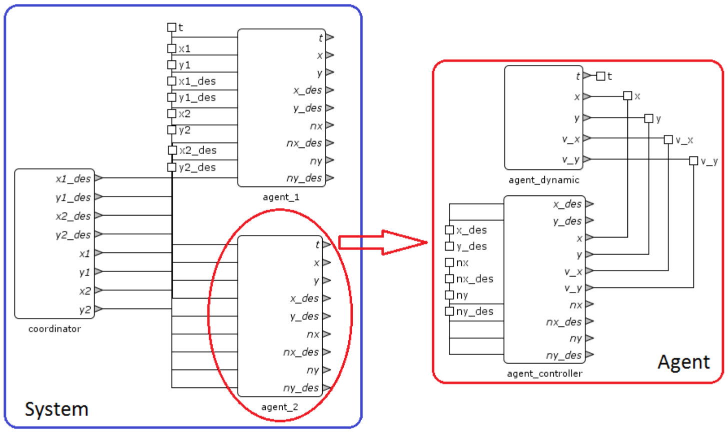


Figure 4.1: *Decentralized-style approach* for modeling autonomous systems using hybrid automata network.

5 Reachability analysis

The benchmark suite including 12 benchmarks ($MAS2 - MAS10_3$) is presented in Table 5.1. In this paper, we present briefly the safety analysis of the benchmark $MAS2$ with two agents. The communication topology of $MAS2$ shows that the robot 2 receive the information from the robot 1. The initial *intermediate positions* of two robots are (x_{h1}^0, y_{h1}^0) and (x_{h2}^0, y_{h2}^0) where $(0 \leq x_{h1}^0 \leq 0.2, 0 \leq y_{h1}^0 \leq 0.1)$ and $(0 \leq x_{h2}^0 \leq 0.2, 0.9 \leq y_{h2}^0 = 1)$. Assume that the distances between the intermediate positions and their corresponding robot centers are $d_1 = d_2 = l = 0.1$. The robots are controlled to go to their *intermediate destinations* $(x_{h1}^d = 3, y_{h1}^d = 3)$ and $(x_{h2}^d = 4, y_{h2}^d = 4)$ while keeping their *intermediate distance* $d_h \geq 1$ as moving. The system is safe if the *distance* between two robots, i.e., between the centers of two robots, is always larger than a threshold $d_{min} = 0.5$. We need to ensure that this threshold is larger than the size of the robots (i.e., the diameter of the robots). The parameters for the distributed control law in Equation 2.4 are chosen as follows: $\alpha_x = 2\alpha_y = 2$, $\gamma_x = 2\gamma_y = 1$.

Figure 5.1 describes the trajectories of the two robots. The figure shows that two robots finally reach their destinations.

Recall that (x_{h1}, y_{h1}) and (x_{h2}, y_{h2}) are not the centers of the robots as given in Equation 2.2. To verify the system safety, let $dis_x = x_2 - x_1$, $dis_y = y_2 - y_1$, $dis_{xh} = x_{h2} - x_{h1}$, $dis_{yh} = y_{h2} - y_{h1}$, the unsafe region of the system can be defined by the following circle.

$$|dis_x|^2 + |dis_y|^2 \leq d_{min}^2$$

If the unsafe region can not be reached, then two robots are always far away from each other at a distance $d > d_{min}$ and then, we can conclude that the system is safe. From Equation 3.3, the reachable sets of dis_x and dis_y can be derived by *bloating* the reachable sets of dis_{xh} and dis_{yh} using the following constraints.

$$x_{h2} - x_{h1} - 2l \leq x_2 - x_1 \leq x_{h2} - x_{h1} + 2l$$
$$y_{h2} - y_{h1} - 2l \leq y_2 - y_1 \leq y_{h2} - y_{h1} + 2l$$

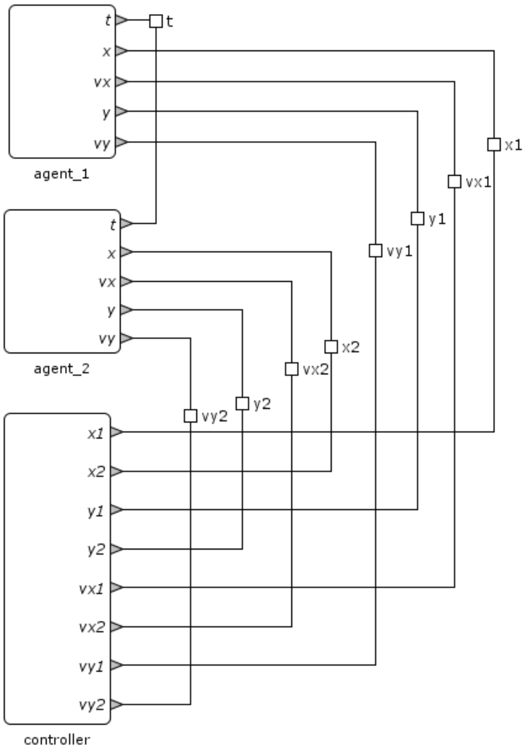


Figure 4.2: *Centralized-style approach* for modeling autonomous systems using hybrid automata network.

Figure 5.2 and Figure 5.3 illustrate the reachable set of dis_{xh} and dis_{yh} over times and Figure 5.4 describes the reachable set of (dis_x, dis_y) (the green polygon) which is bloated from the reachable set of (dis_{xh}, dis_{yh}) (the green polygon). The later figure shows that (dis_x, dis_y) does not reach the unsafe region for all times when the robots move to their destinations. Thus, we can conclude that the system is safe. In addition, we can see that the formation control law actually works since it drives the robots to their destinations and preserve the formation of the robots when they are moving (i.e., the intermediate distance between the robots finally converge to $d_h = \sqrt{2}$.

It is worth noticing that the control parameters $\{\alpha_x, \alpha_y, \gamma_x, \gamma_y\}$ assigned in Equation 2.4 affects significantly the performance of the system. As analyzed in [18], there exists conditions for the control parameters and the communication topology to guarantee that the robots can finally reach their destination while preserving their formation. An appropriate choices of the control parameters can be given from these conditions. In addition, the initial condition and the destination requirements (i.e., the destination positions of the robots) also affects the safety property of the system. For example, if the destination requirements conflict with the formation, the collision may occur.

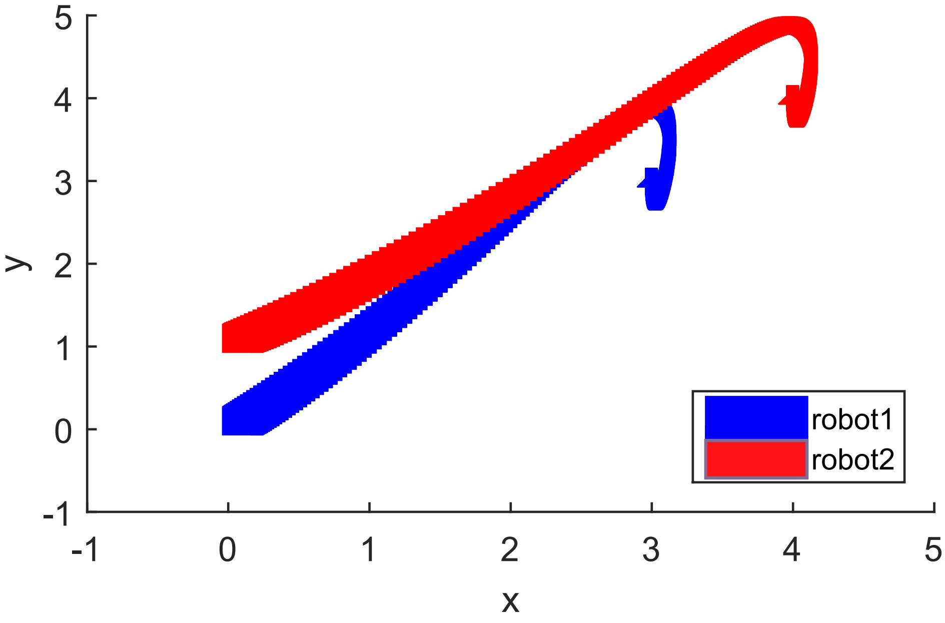


Figure 5.1: Trajectories of the two robots.

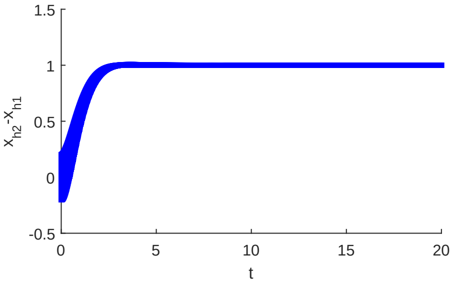


Figure 5.2: Reachable set of $dis_{xh} = x_{h2} - x_{h1}$ over times

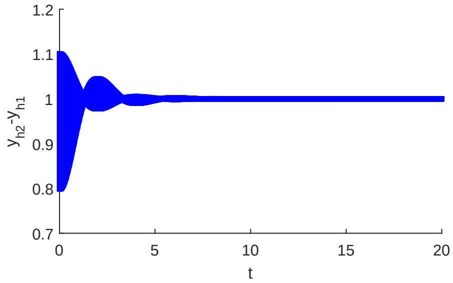


Figure 5.3: Reachable set of $dis_{yh} = y_{h2} - y_{h1}$ over times

6 Outlook

Overall, we present in this paper a set benchmarks for distributed autonomous systems, modeled as network of hybrid automata in the SpaceEx model format. The number of the agents range from two to ten. The position-based formation control has been successfully verified in a benchmark with two agents. There are two important issues should be considered in future

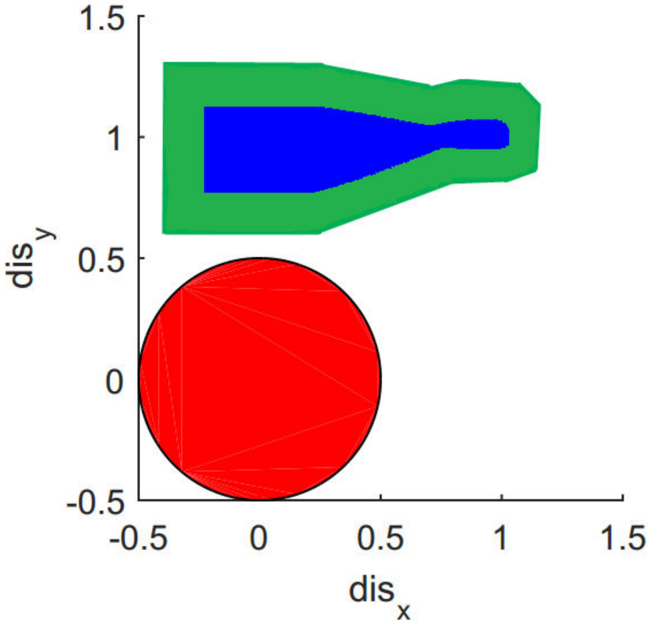


Figure 5.4: Reachable set of (dis_x, dis_y) (the green polygon), (dis_{xh}, dis_{yh}) (the blue polygon) and the unsafe region (inside the red circle).

work. First, it is challenging to model and verify the safety and livenesss properties of the distributed autonomous systems controlled by complex nonlinear formation control laws to avoid collision and obstacles. We can take advantages of verification tools supporting nonlinear hybrid systems such as Flow* [17] and C2E2 [20] in this case. Second, it would be useful for testing verification tools if we can generate automatically distributed autonomous systems with arbitrary large number of agents. We are going to implement this feature as an automatic generator in Hyst [21].

Acknowledgements The material presented in this paper is based upon work supported by the National Science Foundation (NSF) under grant numbers CNS 1464311 and CCF 1527398, the Air Force Research Laboratory (AFRL) through contract number FA8750-15-1-0105, and the Air Force Office of Scientific Research (AFOSR) under contract number FA9550-15-1-0258. The U.S. government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of AFRL, AFOSR, or NSF.

References

[1] H. R. Pulliam, "On the advantages of flocking," *Journal of theoretical Biology*, vol. 38, no. 2, pp. 419–422, 1973.

[2] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," *ACM SIGGRAPH computer graphics*, vol. 21, no. 4, pp. 25–34, 1987.

[3] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[4] R. Olfati-Saber, “Flocking for multi-agent dynamic systems: Algorithms and theory,” *IEEE Transactions on automatic control*, vol. 51, no. 3, pp. 401–420, 2006.

[5] T. Balch and R. C. Arkin, “Behavior-based formation control for multirobot teams,” *IEEE transactions on robotics and automation*, vol. 14, no. 6, pp. 926–939, 1998.

[6] W. Ren and R. W. Beard, “Consensus seeking in multiagent systems under dynamically changing interaction topologies,” *IEEE Transactions on automatic control*, vol. 50, no. 5, pp. 655–661, 2005.

[7] F. Xiao, L. Wang, J. Chen, and Y. Gao, “Finite-time formation control for multi-agent systems,” *Automatica*, vol. 45, no. 11, pp. 2605–2611, 2009.

[8] M. Egerstedt and X. Hu, “Formation constrained multi-agent control,” *IEEE transactions on robotics and automation*, vol. 17, no. 6, pp. 947–951, 2001.

[9] S. Mastellone, D. M. Stipanović, C. R. Graunke, K. A. Intlekofer, and M. W. Spong, “Formation control and collision avoidance for multi-agent non-holonomic systems: Theory and experiments,” *The International Journal of Robotics Research*, vol. 27, no. 1, pp. 107–126, 2008.

[10] A. Rahmani, M. Ji, M. Mesbahi, and M. Egerstedt, “Controllability of multi-agent systems from a graph-theoretic perspective,” *SIAM Journal on Control and Optimization*, vol. 48, no. 1, pp. 162–186, 2009.

[11] K.-K. Oh, M.-C. Park, and H.-S. Ahn, “A survey of multi-agent formation control,” *Automatica*, vol. 53, pp. 424–440, 2015.

[12] T. Johnson and S. Mitra, “Safe flocking in spite of actuator faults,” in *Symposium on Self-Stabilizing Systems*. Springer, 2010, pp. 588–602.

[13] L. Bobadilla, T. T. Johnson, and A. LaViers, “Verified planar formation control algorithms by composition of primitives,” in *AIAA Guidance, Navigation, and Control Conference*, 2015, p. 1541.

[14] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, “Spaceex: Scalable verification of hybrid systems,” in *Computer Aided Verification*. Springer, 2011, pp. 379–395.

[15] S. Kong, S. Gao, W. Chen, and E. Clarke, “dreach: δ-reachability analysis for hybrid systems,” pp. 200–205, 2015.

[16] S. Bak, “Hycreate: A tool for overapproximating reachability of hybrid automata,” *Retrieved January*, vol. 17, p. 2016, 2013.

[17] X. Chen, E. Ábrahám, and S. Sankaranarayanan, “Flow*: An analyzer for non-linear hybrid systems,” in *International Conference on Computer Aided Verification*. Springer, 2013, pp. 258–263.

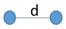
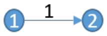
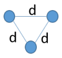
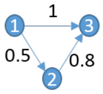
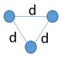
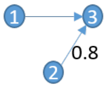
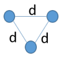
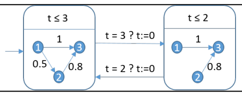
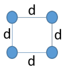
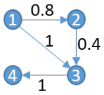
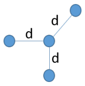
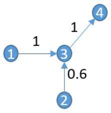
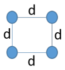
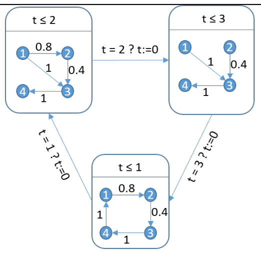
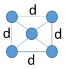
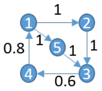
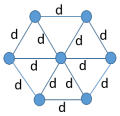
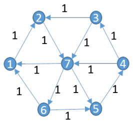
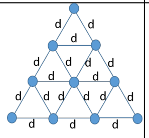
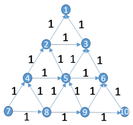
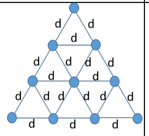
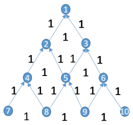
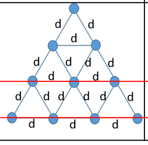
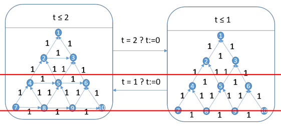
[18] W. Ren and E. Atkins, “Distributed multi-vehicle coordinated control via local information exchange,” *International Journal of Robust and Nonlinear Control*, vol. 17, no. 10-11, pp. 1002–1033, 2007.

[19] H.-D. Tran, L. V. Nguyen, W. Xiang, and T. T. Johnson, “Order-reduction abstractions for safety verification of high-dimensional linear systems,” *Discrete Event Dynamic Systems, Special Issues on Formal Method in Control, to appear*, 2017.

[20] P. S. Duggirala, S. Mitra, M. Viswanathan, and M. Potok, “C2e2: a verification tool for stateflow models,” in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2015, pp. 68–82.

[21] S. Bak, S. Bogomolov, and T. T. Johnson, “Hyst: a source transformation and translation tool for hybrid automaton models,” in *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*. ACM, 2015, pp. 128–133.

Table 5.1: Benchmark collection

No.	Benchmarks	n	Formation	Communication Topology
1	$MAS2$	2		
2	$MAS3_1$	3		
3	$MAS3_2$	3		
4	$MAS3_3$	3		
5	$MAS4_1$	4		
6	$MAS4_2$	4		
7	$MAS4_3$	4		
8	$MAS5$	5		
9	$MAS7$	7		
10	$MAS10_1$	10		
11	$MAS10_2$	10		
12	$MAS10_3$	10		

Discrete Event Dynamic Systems. manuscript No.
(will be inserted by the editor)

Order-Reduction Abstractions for Safety Verification of High-Dimensional Linear Systems

Hoang-Dung Tran, Luan Viet Nguyen,
Weiming Xiang, Taylor T. Johnson

Received: February 15, 2016; Revised: August 31, 2016 and February 20, 2017 / Accepted: date

Abstract Order-reduction is a standard automated approximation technique for computer-aided design, analysis, and simulation of many classes of systems, from circuits to buildings. To be used as a sound abstraction for formal verification, a measure of the similarity of behavior must be formalized and computed, which we develop in a computational way for a class of asymptotic stable linear systems as the main contributions of this paper. We have implemented the order-reduction as a sound abstraction process through a source-to-source model transformation in the HyST tool and use SpaceEx to compute sets of reachable states to verify properties of the full-order system through analysis of the reduced-order system. Our experimental results suggest systems with thousand of state variables can be reduced to systems with tens of state variables such that the order-reduction overapproximation error is small enough to prove or disprove safety properties of interest using current reachability analysis tools. Our results illustrate this approach is effective in tackling the state-space explosion problem for verification of high-dimensional linear systems.

Keywords Abstraction; model reduction; order reduction; verification; reachability analysis

1 Introduction

The state-space explosion problem is a fundamental challenge in model checking and automated formal verification that has received significant attention from the verification community. Among many solutions, abstractions based on the concepts of exact and approximate simulation and bisimulation relations have proved to be effective approaches for obtaining smaller state spaces by abstracting away information that is not needed in the verification process. Such abstractions have been applied

H.D. Tran, W. Xiang, and T.T. Johnson
Vanderbilt University, USA

L.V. Nguyen
University of Texas at Arlington, USA

broadly to simplify the controller synthesis and safety verification process for complex systems. Exact bisimulation relation-based abstractions for safety verification and controller synthesis have been investigated widely in the last decade [Pappas (2003); van der Schaft (2004); Tanner and Pappas (2003); Tabuada and Pappas (2004)]. In this context, the outputs of the abstract system capture exactly the outputs of the original system. As pointed out in [Girard et al. (2008); Girard and Pappas (2007)], the term "exact" is not adequate when dealing with continuous and hybrid systems observed over real numbers since there may be numerical errors in observation, noise, and other imperfections. To obtain an abstraction that guarantees more robust relationship between systems, approximate bisimulation has been proposed and studied extensively in recent years [Girard et al. (2008); Girard and Pappas (2007); Julius (2006); Girard et al. (2006); Islam et al. (2015)]. The main advantage of this approach is that they allow a bounded precision δ which describes how "far off" the executions of the abstraction may be from those of the original system. Then, verifying whether the executions of the original system reach an unsafe region U can be reduced to verify whether the executions of the abstraction (with much lower dimension) reach the δ-neighborhood of the unsafe region U. Thus, finding an efficient way of computing a tight bound on the precision becomes an essential task for this approach. The proposed method shows a great benefit when it can deal both stable and unstable systems. Balanced truncation model reduction, a well-known technique in control [Antoulas et al. (2001)], has recently been applied to obtain abstractions for formal verification of continuous and hybrid systems [Han and Krogh (2004); Han (2005)]. The bounded precision between a system and its abstraction, which is also essential in this framework, is determined using simulation.

Inspired by the results reported in [Girard and Pappas (2007); Han and Krogh (2004)], in this paper, we discuss and improve the computation frameworks in existing techniques to make them more applicable to practical high-dimensional systems. The main contribution of our work is improving the way of computing and using the precision δ. Our method is shown to be efficient, robust and scalable compared with existing approaches over a set of practical benchmarks (several to a thousand dimensions). We implement the method as a model transformation pass within the HyST model transformation tool [Bak et al. (2015)] which enables us to easily apply the techniques and compare results.

The remainder of the paper is organized as follows. Section 2 gives essential definitions used throughout the paper. Section 3 presents methods to find output abstractions of the linear time invariant (LTI) systems using the balanced truncation model reduction method. Section 4 discusses how to verify safety properties for a full-order LTI system using its output abstraction. Section 5 describes our implementation of the method in a prototype tool, and presents a number of examples to illustrate and evaluate the benefits of our method.

2 Preliminaries

Consider two *asymptotically stable* LTI systems:

$$M_n : \begin{cases} \dot{x}(t) = Ax(t) + Bu(t) \\ y(t) = Cx(t). \end{cases} \qquad M_k : \begin{cases} \dot{x}_r(t) = A_r x_r(t) + B_r u(t), \\ y_r(t) = C_r x_r(t). \end{cases}$$

where $x(t) \in \mathbb{R}^n$, $x_r(t) \in \mathbb{R}^k$ are the *system states*; $y(t) \in \mathbb{R}^p$, $y_r(t) \in \mathbb{R}^p$ are the *system outputs*; $u(t)$ is the *control input*; $A \in \mathbb{R}^{n\times n}$, $A_r \in \mathbb{R}^{k\times k}$, $B \in \mathbb{R}^{n\times m}$, $B_r \in \mathbb{R}^{k\times m}$ and $C \in \mathbb{R}^{p\times n}$, $C_r \in \mathbb{R}^{p\times k}$ are system matrices. The initial set of states of M_n and M_k are respectively denoted by $X_0(M_n) \subseteq \mathbb{R}^n$, $X_0(M_k) \subseteq \mathbb{R}^k$ and the set of control inputs is denoted by U. We write initial conditions as $x(0) \in X_0(M_n)$, $x_r(0) \in X_0(M_k)$ and the control input as $u(s) \in U, \forall s \in [0,t]$. In this paper, we assume that M_n is observable and controllable.

Definition 1 Output Abstraction. M_k is called a *k-dimensional output abstraction of M_n* with *precision* $\delta = [\delta_1, \delta_2, \ldots, \delta_p]^T$, denoted as M_k^δ, where each δ_i is a finite positive real if, for $\forall x(0) \in X_0(M_n)$ and $u \in U$, $\exists x_r(0) \in X_0(M_k)$ such that, $\forall t \geq 0$, $\left\|y^i(t) - y_r^i(t)\right\| \leq \delta_i$, $1 \leq i \leq p$ where $y^i(t)$ is the i^{th} component of the output y at time t, and $\|\cdot\|$ denotes the Euclidean norm.

Informally, the output abstraction *behaviors*, which are defined as the trajectories of the output over real time intervals, will approximate within δ the *behaviors* of the full-order system M_n for all time.

Definition 2 Output Reach Set [Han and Krogh (2004)]. The output reach sets *at a time instant* t and *over a time interval* $[0,t_f]$, $t_f \geq 0$ of M_n respectively are:

$$R_t(M_n) \triangleq \{y(t,u,x(0)) | y(t,u,x(0)) = Ce^{At}x(0) + \int_0^t Ce^{A(t-\tau)}Bu(\tau)d\tau\},$$
$$R_{[0,t_f]}(M_n) \triangleq \bigcup\nolimits_{t\in[0,t_f]} R_t(M_n), \text{ where } x(0) \in X_0(M_n),\ u(\tau) \in U, \forall \tau \in [0,t].$$

Definition 3 Safety Specification. A *safety specification* $S(M_n)$ of an LTI system M_n formalizes the safety requirements for the output y of M_n, and is a predicate over the output y of M_n. Formally, $S(M_n) \subseteq \mathbb{R}^p$. The *dual unsafe specification* $U(M_n)$ of the system is also a predicate over the system output, i.e. $U(M_n) \subseteq \mathbb{R}^p$.

Definition 4 Safety Verification. The *time-bounded safety verification problem* is to verify whether the system M_n satisfies a safety specification $S(M_n)$ over an *interval of time* $[0,t_f]$ *(t_f is finite and positive)*, which is described formally in terms of the output reach set as:

$$R_{[0,t_f]}(M_n) \cap \neg S(M_n) = \emptyset \Leftrightarrow M_n \vDash S(M_n),$$
$$R_{[0,t_f]}(M_n) \cap \neg S(M_n) \neq \emptyset \Leftrightarrow M_n \nvDash S(M_n).$$

If M_n satisfies $S(M_n)$, then it is *safe* and we write $M_n \vDash S(M_n)$. If M_n does not satisfy $S(M_n)$, then it is *unsafe* and we write $M_n \nvDash S(M_n)$. The notation "$\neg$" denotes the logical negation which corresponds to set complement in the formulas above.

Definition 5 Safety Specification Transformation. The *safety specification transformation* is the process of finding the corresponding safety (or dually, unsafe) specification for the output abstraction M_k^δ denoted by $S(M_k^\delta) \in \mathbb{R}^p$ (and dually $U(M_k^\delta) \in \mathbb{R}^p$) from the safety specification $S(M_n)$ of the full-order system M_n to guarantee the safety relation defined by:

$$\begin{aligned} R_{[0,t_f]}(M_k^\delta) \cap \neg S(M_k^\delta) = \emptyset \Rightarrow M_n \vDash S(M_n), \\ R_{[0,t_f]}(M_k^\delta) \cap U(M_k^\delta) \neq \emptyset \Rightarrow M_n \nvDash S(M_n). \end{aligned} \tag{1}$$

The main objective of this paper is to perform the safety verification for high-dimensional LTI systems using output abstraction and its transformed safety specification to reduce the computational complexity.

3 Output Abstractions from Balanced Truncation Reduction

The balanced truncation model reduction (BTMR) [Moore (1981)] is a well-known method in control which is used to find a reduced model M_k for an asymptotically stable large scale linear system M_n so that its output trajectory captures the output trajectory of the original system within a estimated bounded error under *an identity control input*. The first step in BTMR is to implement a balanced transformation $\tilde{x}(t) = Hx(t),\ H \in \mathbb{R}^{n\times n}$ to transform M_n to an equivalent balanced system $\tilde{M}_n$. The k-order reduced model M_k which is also asymptotically stable is then obtained by selecting the first k states in the state vector of $\tilde{M}_n$ and truncating the other $n-k$ states (i.e. $x_r = S\tilde{x}(t) = SHx(t),\ S = \begin{pmatrix} I_{k\times k} & 0_{k\times(n-k)} \end{pmatrix}$). The $(n+k)$-dimensional (asymptotically stable) augmented system $\bar{M}_{n+k}$ is defined by:

$$\dot{\bar{x}} = \bar{A}\bar{x} + \bar{B}u = \begin{pmatrix} \tilde{A} & 0 \\ 0 & A_r \end{pmatrix}\bar{x} + \begin{pmatrix} \tilde{B} \\ B_r \end{pmatrix}u,$$
$$\bar{y} = \bar{C}\bar{x} = \begin{pmatrix} \tilde{C} & -C_r \end{pmatrix}\bar{x},$$

where $\bar{x} = \begin{pmatrix} \tilde{x} \\ x_r \end{pmatrix}$; $\tilde{A}$, $\tilde{B}$ and $\tilde{C}$ are the matrices of balanced system $\tilde{M}_n$.

To derive an output abstraction M_k^δ from the system M_n, we first use BTMR to obtain the reduced model M_k, and then determine the precision δ which relates not only to the control input $u(t)$ but also to the initial condition $x(0)$. Determining the precision δ is equivalent to determining the bounds of the augmented system's individual outputs. Let $e_1(t) = \bar{C}e^{\bar{A}t}\bar{x}(0)$ be the *zero input response* and $e_2(t) = \int_0^t \bar{C}e^{\bar{A}(t-\tau)}\bar{B}u(\tau)d\tau$ be the *zero state response*. It is easy to see that $\left\|y^i(t) - y_r^i(t)\right\| = \left\|\bar{y}^i(t)\right\| \leq \left\|e_1^i(t)\right\| + \left\|e_2^i(t)\right\| = \delta_i$, where $h^i(t)$ denotes the i^{th} element of vector $h(t)$.

The following theorems obtain the theoretical bounds for $e_1(t)$ and $e_2(t)$. The sum of these two bounds gives us the precision δ.

Theorem 1 *Let $\bar{x}(0) = \left(Hx(0)\ SHx(0)\right)^T$, then the zero input response $e_1(t)$ of the asymptotically stable augmented system $\bar{M}_{n+k}$ satisfies the following inequality for all $t \in \mathbb{R}_{\geq 0}$:*

$$\left\|e_1^i(t)\right\| \leq \left\|\bar{C}_i\right\| \cdot \sup_{x(0)\in X_0} \left\|\bar{x}(0)\right\|,\ 1 \leq i \leq p,$$

where $\mathbb{R}_{\geq 0}$ is the set of non-negative real numbers, $\bar{C}_i$ is the row i of the matrix $\bar{C}$.

The proof of Theorem 1 is given in Appendix 7.1.

Theorem 2 *Let $\bar{x}(0) = \left(Hx(0)\ SHx(0)\right)^T$ and $P_0 > 0$ is the solution of the following optimization problem:*

$$P_0 = min(trace(P))\ subject\ to$$
$$P > 0,\ \bar{A}^TP + P\bar{A} < 0,\ \bar{C}_i^T\bar{C}_i \leq P$$

where $\bar{C}_i$ is the row i of the matrix $\bar{C}$. Then, the zero input response $e_1(t)$ of the asymptotically stable augmented system $\bar{M}_{n+k}$ satisfies the following inequality for all $t \in \mathbb{R}_{\geq 0}$:

$$\left\|e_1^i(t)\right\| \leq \sup_{x(0)\in X_0} \sqrt{\bar{x}(0)^TP_0\bar{x}(0)},\ 1 \leq i \leq p,$$

The proof of Theorem 2 is given in Appendix 7.2.

Theorem 3 *[Han and Krogh (2004); Obinata and Anderson (2012)]. The error e_2 between the full-order asymptotically stable system M_n and its k-dimensional reduced system M_k satisfies the following inequality for all $t \in \mathbb{R}_{\geq 0}$:*

$$\left\|e_2^i(t)\right\| \leq (4\sum_{j=k+1}^{n}(2j-1)\sigma_j)\cdot\|u(t)\|_\infty,\ 1 \leq i \leq p,$$

where σ_j is the j^{th} Hankel singular value of the system M_n and $\|u(t)\|_\infty = \sup_{t\in\mathbb{R}_{\geq 0}}\|u(t)\|$

We note that the precision δ can be obtained using different methods all of which have their advantages and drawbacks. In [Han and Krogh (2004)], the authors propose a simulation-based approach to determine δ. To determine the bound of e_1, the authors simulate the full-order system and the reduced system from each vertex in a polyhedral representation of the initial set of states. The advantage in using this method is that it gives a very tight bound of e_1 while the drawback is that the number of simulations required can grow exponentially. For example, if the initial set is a hypercube in 100-dimensions, we have to simulate the full-order system and its reduced system with $2^n = 2^{100}$ vertices, which is infeasible even if each simulation takes little time. The bound of e_2 is determined by integrating the norm of the impulse response of the augmented system via simulation. This method is especially useful in practice since it gives a tight bound for e_2 with only m simulations, where m is the number of inputs. Alternatively, without separately computing the bounds of e_1 and e_2, the precision δ can be calculated by solving a set of LMI optimization problems on sets of initial states and inputs [Girard and Pappas (2007)]. This approach has advantages when dealing with small and medium-dimensional systems (less than 50 dimensions) and works for both stable and unstable systems. When the system dimension is large, the error bound obtained is overly conservative and may not be useful.

Exploiting the advantages and overcoming the drawbacks of existing approaches when dealing with practical systems are the main contributions of our approach. First, we compute the bounds for e_1 and e_2 separately, making the computation process more robust and thus produce less conservative results in comparison with [Girard and Pappas (2007)]. Second, Theorem 1 and Theorem 2 are proposed to overcome the drawback of simulation-based method [Han and Krogh (2004)] as described above. It should be emphasized that in most circumstances, our approach can be used to compute the bound of e_1 and the simulation-based approach can be used to determine the bound of e_2. This combination is scalable for high-dimensional systems while still obtaining a good precision δ. Finally, the output abstraction defined based on individual outputs $(y^i(t), y_r^i(t))$ and precisions δ_i allows us to easily verify the safety of multi-output systems. Next, we briefly analyze the time complexity in computing the precision δ to show theoretically the scalability of different methods.

We analyze the complexity of the simulation-based approach [Han and Krogh (2004)] in terms of number of simulations. For an n-dimension system with m inputs and p outputs, the number of vertices in a polyhedral initial set (in the worst case) is 2^n. Therefore, the number of simulations needed to determine the bound of e_1 is 2^n. For e_2, the number of simulations needed is m. Consequently, the number of simulations needed is $O((2^n+m))$. In [Girard and Pappas (2007)], to determine the precision δ, this method solves two LMIs and quadratic optimization problems on the sets of

[Girard and Pappas (2007)]	$O(n^6)$
[Han and Krogh (2004)]	$O(2^n + m)$
Theorem 1	$O(n^{3.5})$
Theorem 2	$O((n+k)^{5.5})$

Table 1: Time complexity of different methods to compute the precision δ.

initial state and inputs in which the time complexity for solving two LMI constraints using interior point algorithms can be estimated by $O([(n^2+n)/2]^{2.75} \times 2^{1.5})$ [Vandenberghe and Boyd (1994); Nesterov et al. (1994)] and the time complexity for solving the optimization problem in [Girard and Pappas (2007)] using interior point algorithms is $O([(n^2+n)/2]^3)$. Overall, the time complexity of computing the precision is $O([(n^2+n)/2]^3) + O([(n^2+n)/2]^{2.75} \times 2^{1.5})$, which can be bounded by $O(n^6)$. In our approach, it is easy to see that Theorem 1 computation mainly relates to solving the optimization problem to find $\sup_{x(0)\in X_0} \|\bar{x}(0)\|$. The time complexity for solving this problem using interior point algorithms is $O((n+1)^{3.5})$. For Theorem 2, we first need to solve the eigenvalue problem (EVP) subject to two matrix inequalities that has time complexity $O([((n+k)^2+n+k)/2]^{2.75} \times 2^{1.5})$ if using interior point algorithms. Then, we need to solve the quadratic optimization problem that has time complexity $O((n+k)^3)$ (using the interior point algorithm). Overall, the time complexity of Theorem 2 can be bounded by $O((n+k)^{5.5})$. The computational cost of Theorem 3 in our approach is smaller compared to Theorem 1 and Theorem 2. Table 1 shows the simplified time complexity analysis of different approaches. The computation time of these approaches will be measured and discussed in detail in Section 5.

4 Safety Verification with Output Abstractions

It should be emphasized that first, our approach is different from [Han and Krogh (2004)] in the way of using the precision δ where the precision is used to compute the reach set of the full-order system before using this reach set to verify the safety of the system. In our approach, the precision is used to obtain the safety specification of the output abstraction that satisfies the safety relation (1). Then, we verify the safety of the output abstraction to conclude safety of the original system. It is important to notice from (1) that it may be the case that we cannot conclude anything about the safety of the original system using the output abstraction. Second, the output abstraction is different from the approximate abstraction [Girard and Pappas (2007)] which depends on a single precision δ because it is defined based on individual outputs $(y^i(t), y_r^i(t))$ and individual precisions δ_i. As a result, we can get a more accurate safety specification transformation based on the individual precisions. This transformation is convenient for verifying safety in practical multi-output systems where the magnitudes of element outputs are usually significantly different from each other. Our safety transformation rules are addressed in the following.

$S(M_n)$ as Convex Polytopes. Assume that the safety specification of the full-order system is in the following form:

$$S(M_n) = \{y \in \mathbb{R}^p |\ \Gamma y + \Psi \leq 0,\ \Gamma = [\alpha_{ij}] \in \mathbb{R}^{q\times p},\ \Psi = [\beta_i] \in \mathbb{R}^q\}, \qquad (2)$$

Lemma 1 *Given $S(M_n)$ described by (2), then $S(M_k^\delta)$ and $U(M_k^\delta)$ defined as follows guarantee the safety relation (1).*

$$\begin{aligned} S(M_k^\delta) &= \{y_r \in \mathbb{R}^p |\ \Gamma y_r + \overline{\Psi} \leq 0,\ \overline{\Psi} = \Psi + \Delta\}, \\ U(M_k^\delta) &= \{y_r \in \mathbb{R}^p |\ \Gamma y_r + \underline{\Psi} > 0,\ \underline{\Psi} = \Psi - \Delta\}. \end{aligned} \tag{3}$$

where $\Delta = [\Delta_i] \in \mathbb{R}^q,\ \Delta_i = \sum_{j=1}^{p} |\alpha_{ij}|\delta_j$.

The proof is given in Appendix 7.1.

$S(M_n)$ as Ellipsoids. Assume that the safety specification of the full-order system is described by an ellipsoid with radius R and centered at $a \in \mathbb{R}^p$ as:

$$S(M_n) = \{y \in \mathbb{R}^p |\ (y-a)^T Q(y-a) \leq R^2,\ a \in \mathbb{R}^p,\ R > 0,\ 0 < Q \in \mathbb{R}^{p \times p}\}, \tag{4}$$

Since Q is a symmetric matrix, there exists an orthogonal matrix $E = [l_1, l_2, .., l_p]$ $= [\gamma_{ij}] \in \mathbb{R}^{p \times p}$ such that $E^T Q E = \Lambda = diag(\lambda_1,\ \lambda_2, ..., \lambda_p)$, where λ_i (> 0) is eigenvalue of Q and l_i is the eigenvector of Q corresponding to λ_i. The transformed safety and unsafe specifications of the output abstraction can be obtained using the following lemma.

Lemma 2 *Given $S(M_n)$ described by (4), then $S(M_k^\delta)$ and $U(M_k^\delta)$ defined as follows guarantee the safety relation (1):*

$$\begin{aligned} S(M_k^\delta) &= \{y_r \in \mathbb{R}^p |\ (y_r - a)^T Q(y_r - a) \leq (R - \Delta_R)^2\}, \\ U(M_k^\delta) &= \{y_r \in \mathbb{R}^p |\ (y_r - a)^T Q(y_r - a) > (R + \Delta_R)^2\}, \\ \Delta_R &= \sqrt{\sum\nolimits_{i=1}^{p} [\lambda_i (\sum\nolimits_{j=1}^{p} |\gamma_{ij}|\delta_j)^2]}. \end{aligned}$$

The proof of this result is given in Appendix 7.2.

We can see that the transformed safety specification of the output abstraction is also an ellipsoid (with smaller radius $R - \Delta_R$) located inside the original ellipse defining the safety specification of the full-order system. Meanwhile the corresponding transformed unsafe specification is defined by the region outside the larger ellipse with the radius $R + \Delta_R$.

Lemma 3 *Given an asymptotically stable LTI system M_n, whenever its output abstraction M_k^δ is safe or unsafe, it is sound to claim that the system M_n is safe or unsafe respectively.*

Proof According to Section 3, for a stable LTI system M_n, there exists an output abstraction M_k^δ. We can see that, for any control input $u \in U$ and initial state $x(0) \in X_0(M_n)$, there is a set of p output trajectories $y_i, 1 \leq i \leq p$ of M_n. From the definition of the output abstraction, there exists a corresponding initial state $x_r(0) \in X_0(M_k)$ such that the abstraction produces a set of p output trajectories $y_r^i, 1 \leq i \leq p$ in which the distance between two pair trajectories $\left\| y^i - y_r^i \right\|$ is always bounded by a sound δ_i *all the time*. Moreover from Lemmas 1 and 2, because the transformed specifications ($S(M_k^\delta)$ and $U(M_k^\delta)$) satisfy the safety relation (1), thus when the set of p output trajectories $y_r^i, 1 \leq i \leq p$ of M_k^δ satisfies the transformed safety specification $S(M_k^\delta)$, its corresponding set of p output trajectories $y_i, 1 \leq i \leq p$ of M_n also satisfies the original safety specification $S(M_n)$ *all the time*. Consequently, when the output abstraction is safe, it is sound to claim that the full-order system is safe. A similar proof can be given for the unsafe case. This completes the proof.

Remark 1 It is useful to mention that the procedure of safety verification for high-dimensional linear systems using order-reduction abstraction can be done automatically. The core issue is checking whether the reach set of the abstraction returned by verification tools such as SpaceEx is contained inside the transformed safe set. This can be done in two steps by taking advantages of the existing powerful SMT solvers such as Z3 [De Moura and Bjørner (2008)]. First, the transformed safe set can be described by a predicate over the output variables. Then, we check whether the predicate is satisfied by all vertices of the returned reach set from SpaceEx using a SMT solver.

We have studied using output abstraction for safety verification. Next, we evaluate and compare our approach with others via a set of benchmarks.

5 Case Studies and Evaluation

To evaluate the order-reduction abstraction method presented in this paper, we have implemented a software prototype that automatically creates output abstractions from full-order systems and applied it to a set of benchmarks [Chahlaoui and

Benchmark	**Property**	**Initial set of states** $X_0 = \{x_0 \in \mathbb{R}^n \mid lb(i) \leq x_0(i) \leq ub(i),\ 1 \leq i \leq n\}$	**Input constraint** $u = [u_1, \cdots, u_m]^T$	**Safety specification** $y = [y_1, \cdots, y_p]^T$
Motor control system (MCS)	$n = 8$ $m = 2$ $p = 2$	$lb(i) = ub(i) = 0,\ i = 2, 3, 4, 6, 7, 8,$ $lb(2) = 0.002,\ ub(2) = 0.0025,$ $lb(3) = 0.001,\ ub(3) = 0.0015.$	$u_1 \in [0.16,\ 0.3],$ $u_2 \in [0.2,\ 0.4].$	unsafe region: $0.35 \leq y_1 \leq 0.4,$ $0.45 \leq y_2 \leq 0.6.$
Helicopter (HELI) [7]	$n = 28$ $m = 6$ $p = 2$	$lb(i) = ub(i) = 0.1,\ i = 1, 4, 5, 6, 7,$ $lb(2) = lb(3) = 0.098, ub(2) = 0.11, ub(3) = 0.102,$ $lb(i) = ub(i) = 0,\ 8 \leq i \leq 28.$	$u_i \in [-1,\ 1],$ $1 \leq i \leq 6.$	unsafe region: $-1 \leq y_1 \leq 1,$ $10 \leq y_2$
Building model (BM) [4]	$n = 48$ $m = 1$ $p = 1$	$lb(i) = 0.0002,\ ub(i) = 0.00025,\ 1 \leq i \leq 10,$ $lb(25) = -0.0001,\ ub(25) = 0.0001,$ $lb(i) = ub(i) = 0,\ 11 \leq i \leq 48,\ i \neq 25.$	$u_1 \in [0.8,\ 1].$	unsafe region: $0.008 \leq y_1$
Partial differential equation (PDE) [4]	$n = 84$ $m = 1$ $p = 1$	$lb(i) = 0,\ ub(i) = 0,\ 1 \leq i \leq 64$ $lb(i) = 0.001,\ ub(i) = 0.0015,\ 64 \leq i \leq 80,$ $lb(i) = -0.002,\ ub(i) = -0.0015,\ 81 \leq i \leq 84.$	$u_1 \in [0.5,\ 1].$	safe region: $y_1 \leq 12$
International space station (ISS) [4]	$n = 270$ $m = 3$ $p = 3$	$lb(i) = -0.0001,\ ub(i) = 0.0001,\ 1 \leq i \leq 270.$	$u_1 \in [0,\ 0.1],$ $u_2 \in [0.8,\ 1],$ $u_3 \in [0.9,\ 1].$	Safe region: $-461y_1 + 887y_2 + 0.67 \leq 0,$ $-440y_1 - 898y_2 - 0.68 \leq 0,$ $-76.7y_1 + 997y_2 - 0.54 \leq 0,$ $898y_1 - 440y_2 - 0.89 \leq 0,$ $945y_1 + 326y_2 - 0.95 \leq 0,$ $-0.0005 \leq y_3 \leq 0.0005.$
FOM [4]	$n = 1006$ $m = 1$ $p = 1$	$lb(i) = -0.0001,\ ub(i) = 0.0001,\ 1 \leq i \leq 400$ $lb(i) = 0.0002,\ ub(i) = 0.00025,\ 401 \leq i \leq 800,$ $lb(i) = 0,\ ub(i) = 0,\ 801 \leq i \leq 1006.$	$u_1 \in [-1,\ 1].$	safe region: $y_1 \leq 45$

Table 2: Benchmarks for the order-reduction abstraction method in which n is dimension of the system; m and p are the number of inputs and outputs respectively.

	k	Bisim		Sim			Mixed bound				Theoretical bound			
		δ	$t(s)$	δ	$t(s)$	N	e_1	e_2	δ	$t(s)$	e_1	e_2	δ	$t(s)$
MCS	5	2.1	0.93	$\begin{pmatrix}0.0021\\0.047\end{pmatrix}$	0.17	2^2+2	$\begin{pmatrix}0.00049\\0.00062\end{pmatrix}$	$\begin{pmatrix}0.002\\0.047\end{pmatrix}$	$\begin{pmatrix}0.0025\\0.047\end{pmatrix}$	0.92	$\begin{pmatrix}0.0098\\0.0093\end{pmatrix}$	$\begin{pmatrix}0.53\\0.53\end{pmatrix}$	$\begin{pmatrix}0.54\\0.54\end{pmatrix}$	0.15
	4	1.5	0.64	$\begin{pmatrix}0.036\\0.047\end{pmatrix}$	0.19	2^2+2	$\begin{pmatrix}0.00086\\0.00062\end{pmatrix}$	$\begin{pmatrix}0.035\\0.047\end{pmatrix}$	$\begin{pmatrix}0.036\\0.047\end{pmatrix}$	0.9	$\begin{pmatrix}0.009\\0.009\end{pmatrix}$	$\begin{pmatrix}0.91\\0.91\end{pmatrix}$	$\begin{pmatrix}0.92\\0.92\end{pmatrix}$	0.15
HELI	20	0.84	17	$\begin{pmatrix}7.1e-05\\4.3e-05\end{pmatrix}$	0.6	2^4+6	$\begin{pmatrix}0.0072\\0.018\end{pmatrix}$	$\begin{pmatrix}5.7e-05\\3.2e-05\end{pmatrix}$	$\begin{pmatrix}0.0073\\0.018\end{pmatrix}$	35	$\begin{pmatrix}0.28\\0.95\end{pmatrix}$	$\begin{pmatrix}0.0017\\0.0017\end{pmatrix}$	$\begin{pmatrix}0.28\\0.95\end{pmatrix}$	0.49
	16	28	12	$\begin{pmatrix}0.00075\\0.0013\end{pmatrix}$	0.56	2^4+6	$\begin{pmatrix}0.0072\\0.018\end{pmatrix}$	$\begin{pmatrix}0.0007\\0.00087\end{pmatrix}$	$\begin{pmatrix}0.0079\\0.019\end{pmatrix}$	23	$\begin{pmatrix}0.28\\0.95\end{pmatrix}$	$\begin{pmatrix}0.029\\0.029\end{pmatrix}$	$\begin{pmatrix}0.3\\0.97\end{pmatrix}$	0.45
	10	160	8.1	$\begin{pmatrix}0.024\\0.038\end{pmatrix}$	0.55	2^4+6	$\begin{pmatrix}0.0085\\0.021\end{pmatrix}$	$\begin{pmatrix}0.021\\0.031\end{pmatrix}$	$\begin{pmatrix}0.03\\0.053\end{pmatrix}$	13	$\begin{pmatrix}0.27\\0.93\end{pmatrix}$	$\begin{pmatrix}1\\1\end{pmatrix}$	$\begin{pmatrix}1.3\\1.9\end{pmatrix}$	0.45
BM	25	0.0096	180	0.0051	22	$2^{11}+1$	0.013	$6.2e-05$	0.013	130	0.083	0.0072	0.09	1
	15	0.069	120	0.005	18	$2^{11}+1$	0.012	0.00044	0.013	58	0.078	0.084	0.16	0.97
	6	0.1	44	0.0058	14	$2^{11}+1$	0.011	0.00025	0.012	24	0.073	0.21	0.28	0.98
PDE	30	0.75	230	N/A	OOT	$2^{20}+1$	0.033	$5.6e-14$	0.033	1500	1	$5e-12$	1	1.7
	20	0.038	160	N/A	OOT	$2^{20}+1$	0.033	$3.5e-14$	0.033	890	1	$5.4e-12$	1	1.7
	10	0.086	55	N/A	OOT	$2^{20}+1$	0.033	$9.8e-13$	0.033	520	0.92	$2.7e-11$	0.92	1.7
	6	0.1	42	N/A	OOT	$2^{20}+1$	0.033	$3.5e-07$	0.033	370	0.89	$5.5e-06$	0.89	1.7
ISS	25	N/A	OOT	N/A	OOT	$2^{270}+3$	N/A	$\begin{pmatrix}2.1e-05\\0.001\\4.6e-05\end{pmatrix}$	N/A	OOT	$\begin{pmatrix}0.00043\\0.00026\\0.00026\end{pmatrix}$	$\begin{pmatrix}0.47\\0.47\\0.47\end{pmatrix}$	$\begin{pmatrix}0.47\\0.47\\0.47\end{pmatrix}$	11
	10	N/A	OOT	N/A	OOT	$2^{270}+3$	N/A	$\begin{pmatrix}2.4e-05\\5.6e-05\\9e-05\end{pmatrix}$	N/A	OOT	$\begin{pmatrix}0.00042\\0.00022\\0.00021\end{pmatrix}$	$\begin{pmatrix}1.7\\1.7\\1.7\end{pmatrix}$	$\begin{pmatrix}1.7\\1.7\\1.7\end{pmatrix}$	12
FOM	20	N/A	OOT	N/A	OOT	$2^{800}+1$	N/A	$2.7e-07$	N/A	OOT	1.3	$1.1e-05$	1.3	48
	15	N/A	OOT	N/A	OOT	$2^{800}+1$	N/A	0.00021	N/A	OOT	1.3	0.0065	1.3	48
	10	N/A	OOT	N/A	OOT	$2^{800}+1$	N/A	0.1	N/A	OOT	1.3	2.2	3.5	48

Table 3: Experimental results obtained from different methods in which: k is the dimension of the output abstraction, δ is the precision, e_1 is the zero input response error, e_2 is the zero state response error, t is the error computing time (in second) and N is the number of simulations. **Bisim** column contains the results of approximate bisimulation approach proposed by [Girard and Pappas (2007)]. **Sim** column are the results of the simulation-based approach proposed by [Han and Krogh (2004)].

Van Dooren (2005); Frehse et al. (2011)] presented Table 2. The method is integrated in HyST by calling Matlab related functions.[1]

Precision and computation time evaluation. The experiments were performed using Matlab 2014a and SpaceEx [Frehse et al. (2011)] on a personal computer with the following configuration: Intel (R) Core(TM) i7-2677M CPU at 1.80GHz, 4GB RAM, and 64-bit Window 7. We set the upper limit for Matlab simulation and SpaceEx running time at two hours. In all experimental result tables, the term of “N/A” stands for *not applicable* and “OOT” stands for *timeout* when the result could not be computed within two hours.

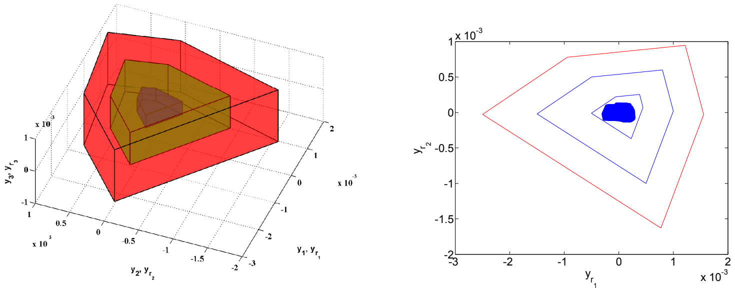
Table 3 presents the precision δ and computation times of the different methods. In the table, the “mixed bound” column contains the bound of e_1 computed by Theorem 2 and the bound of e_2 determined by simulation and the corresponding precision δ. The “theoretical bound” column contains the bounds of e_1 and e_2 which are computed using Theorem 1 and Theorem 3 respectively. The table shows

[1] The prototype implementation and SpaceEx model files for the examples evaluated, both before and after order reduction, are available at: http://verivital.com/hyst/pass-order-reduction/.

Benchmark	**Full Order System**		**Output Abstraction**				
	Time(s)	**Memory(Kb)**	k	$T_1(s)$	$T_2(s)$	**Total time(s)**	**Memory(Kb)**
Motor control system	27	3048	5	26	0.92	26.9	3044
			4	16.7	0.9	17.6	3044
Helicopter	287	3052	20	206	35	241	3052
			16	128	23	151	3048
			10	68	13	81	3048
Building model	893	3056	25	237.2	130	367.2	3048
			15	82.3	58	140.3	3044
			6	19.5	24	43.5	3040
Partial differential equation	OOT	N/A	30	725.6	1500	2225.6	3048
			20	310	890	1200	3048
			10	75.2	520	595.2	3040
			6	31.9	370	401.9	3040
International space station	OOT	N/A	25	254.3	11	265.3	3064
			10	72.8	12	84.8	3052
FOM model	OOT	N/A	20	95.4	48	143.4	3048
			15	56.2	48	104.2	3044
			10	34.8	48	82.8	3040

Table 4: Computation cost for verification process of the full order original LTI system and its output abstractions using SpaceEx in which T_1 is the time for SpaceEx to compute the reach set of the output abstraction; T_2 is the time for obtaining the output abstraction; "Total Time" column states for the total time of verification process for the output abstraction, "Memory" column presents the memory used for computing reach set which is measure in kilobyte; time is measured in second.

that, when the initial set of states X_0 is close to the origin, e.g. BM benchmark, the bounds of e_1 computed by Theorem 1 are fairly good and acceptable. However, conservative results are derived when the initial set of states X_0 is far from the origin, e.g. helicopter benchmark. Theorem 3 indicates that the theoretical bound of e_2 depends on Hankel singular values. This can be seen from the PDE benchmark where the theoretical bounds of e_2 for all cases of the output abstraction's dimension k are very small due to the fact that the Hankel singular values σ_k (which are not presented here) of the corresponding balanced system are very small (almost equal to zero) as $k \geq 5$. In contrast, in the helicopter benchmark, the theoretical bound of e_2 becomes larger when the lower dimension output abstraction is obtained. It is small when $k = 20$ because $\sum_{21}^{28} \sigma_j$ is small. The theoretical bound of e_2 becomes conservative as $k = 10$ since $\sum_{11}^{28} \sigma_j$ is large. From the results in the table, we see that for *low and medium-dimensional systems* ($n < 100$), the simulation-based approach [Han and Krogh (2004)] gives very tight bounds for the errors, e.g, the motor control system and helicopter benchmarks. This approach is powerful when dealing with systems having a small number of vertices in the initial set. When the number of vertices increases (e.g, PDE benchmark), it is unable to apply the simulation-based approach due to the number of simulations growing exponentially. The approximate bisimulation approach [Girard and Pappas (2007)] integrated in the Matisse toolbox gives a good precision for the PDE benchmark but very conservative results for the MCS and Helicopter benchmarks due to the appearance of ill-conditioned matrices in solving LMI and optimization problems. Combining Theorem 2 and simulation bound of e_2 (i.e. mixed bound) is more efficient as it produces very tight precisions for all benchmarks. In the case of high-dimensional systems ($n > 100$) , the approximate bisimulation and simulation-based approaches give no result due to running out of time while our method can still be applied. We can see that Theorem 1 and



(a) Safety specification S_{270} of the full order ISS system which is the region inside the middle green polytopes and the transformed safety specifications of its 10-order output abstraction in which the safe region S_{10}^{δ} is inside the smallest blue polytopes and the unsafe region U_{10}^{δ} is outside the largest red polytope.

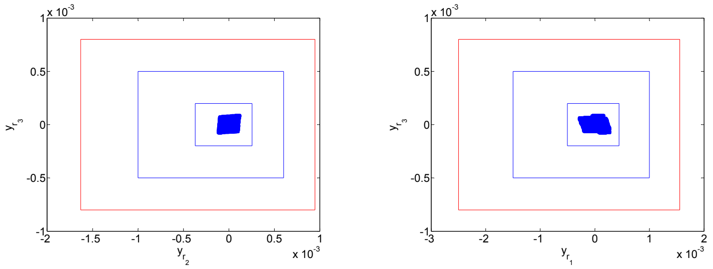
(b) Reachable set of (y_{r1}, y_{r2}) of the ISS's 10-order abstraction M_{10}^{δ} in the period of time $[0,\ 20s]$ and its safe region S_{12}^{δ} (inside the smallest blue polygon) and unsafe region U_{12}^{δ} (outside the red polygon).

Fig. 1: The transformed safety specifications of the 10-order output abstraction of ISS system and the its projected reachable set on the (y_{r_1}, y_{r_2}) plane.

Theorem 3 have small computation times while Theorem 2 and the approximate bisimulation approach require much more time to compute the precision.

Table 4 shows the computation cost of the verification process for the full-order benchmarks and their different output abstractions. The bounded times for running all SpaceEx models are set at $t_f = 20s$. As shown in the table, although using output abstraction does not help much to reduce the memory used in verification, it can help to reduce significantly the computation time. Moreover, output abstraction can be applied to check the safety of high-dimensional systems (e.g. PDE, ISS and FOM) that cannot be verified directly using existing verification tools. Next, we consider the whole process of using output abstraction to verify the safety of the international space station system.

International Space Station (ISS). Verification for the full-order system with 270 state variables (denoted by M_{270}) may be difficult for existing verification tools. Our approach can help to verify safety of such high-dimensional system with a small computation cost. There are different output abstractions that can be used to verify whether the full-order system satisfies its safety requirements. In this paper, a 10-order output abstraction is chosen to check the safety of the full-order system. The precision $\delta = 10^{-3} \times [0.44,\ 0.28,\ 0.3]^T$ between the full-order system and its 10-order output abstraction (denoted by M_{10}^{δ}) is obtained from the theoretical bound of e_1 and the simulation bound of e_2. The safety specification of the full-order ISS system S_{270} is visualized by the region inside the middle blue polytope in Figure 1a. The transformed safety specifications (safe and unsafe specifications) of the corresponding 10-order output abstraction respectively are the region inside the smallest blue polytope and the region outside the red polytope. Figures 1b, 2a and 2b present the safety specification transformation and output reach set in the period of time $[0,\ 20s]$ computed by SpaceEx for the 10-order output abstraction on 2-dimension axes. In the figures, the regions inside the middle blue polygons are the 2-dimensional pro-



(a) Reachable set of (y_{r_2}, y_{r_3}) of the ISS's 10-order abstraction M_{10}^{δ} in the period of time $[0,\ 20s]$ and its safe region S_{23}^{δ} (inside the smallest blue polygon) and unsafe region U_{23}^{δ} (outside the red polygon).

(b) Reachable set of (y_{r_1}, y_{r_3}) of the ISS's 10-order abstraction M_{10}^{δ} in the period of time $[0,\ 20s]$ and its safe region S_{13}^{δ} (inside the smallest blue polygon) and unsafe region U_{13}^{δ} (outside the red polygon).

Fig. 2: The projected transformed safety specifications and reachable set of the 10-order output abstraction of ISS system on the (y_{r_2}, y_{r_3}) and (y_{r_1}, y_{r_3}) planes.

jections of the safety regions of the full-order system. The corresponding projected transformed safety and unsafe specifications S_{10}^{δ}, U_{10}^{δ} of the output abstraction are described by the regions inside the smallest blue polygons and the regions outside the red polygons respectively. The reach sets $R_{ij}^{\delta},\ i \neq j,\ 1 \leq i, j \leq 3$ for each pair output (y_{r_i}, y_{r_j}) of the abstraction M_{10}^{δ} are depicted by the solid blue regions. As shown in Figure 1a, the reach set of the abstraction is completely inside its transformed safe set. Thus, it can be concluded that the full-order system M_{270} satisfies the safety requirement S_{270}. Therefore, the full-order system is safe. The conclusion about the safety of the full-order system can also be given using projection as follows. Due to the fact that one face of the polytopes defining the transformed safe set is aligned with one sub-coordinate axes (i.e. $y_{r_1}Oy_{r_2}$), so for all (i, j), we have $R_{ij}^{\delta} \cap \neg S_{10}^{\delta} = \emptyset \Rightarrow M_{10}^{\delta} \vDash S_{10}^{\delta}$. Consequently, the full-order system is safe. It should be noted that the above relation is not true in general because we cannot conclude a set is a subset of another set by considering the relations of their projections on all sub-coordinate axes. The conclusion about safety of the ISS system is a special case as one face of the safe set is aligned with one sub-coordinate axes.

6 Conclusion and Future Work

We have proposed an approach to verify safety specifications for high-dimensional linear systems by verifying transformed safety specifications of a lower-dimensional output abstraction using existing hybrid system verification tools. By reducing the dimensionality, our method significantly reduces the time of reachability computations in the verification process.

There are two interesting directions for future work. First, the proposed method which only deals with stable linear systems can be extended for unstable linear systems. Second, our approach can also be extended to more general hybrid systems. The main idea is that the states in each location that are related to guards/invariants need to be declared as the outputs of that location. Then, the output abstraction for each location can be obtained. A new hybrid system is then constructed based on these output abstractions. The guards/invariants of the new hybrid system are obtained by transforming the former guards/invariants of the original hybrid system in the same manner of safety specifications transformation proposed in this paper. This approach may benefit from other notions of "similarity" between behaviors (executions) of systems such as discrepancy functions [Duggirala et al. (2013)], or conformance degree [Abbas et al. (2014)].

Acknowledgment

The authors gratefully acknowledge the detailed feedback provided by the reviewers, which have helped improve this manuscript. The authors thank Dr. Andrew Sogokon of Vanderbilt University for carefully reading and providing feedback on the final draft of the manuscript. The material presented in this paper is based upon work supported by the National Science Foundation (NSF) under grant numbers CNS 1464311 and SHF 1527398, the Air Force Research Laboratory (AFRL) through contract number FA8750-15-1-0105, and the Air Force Office of Scientific Research (AFOSR) under contract numbers FA9550-15-1-0258 and FA9550-16-1-0246. The U.S. government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of AFRL, AFOSR, or NSF.

References

Abbas, Haider, Hans Mittelmann, and Georgios Fainekos. 2014. Formal property verification in a conformance testing framework. In *Formal methods and models for codesign (memocode), 2014 twelfth acm/ieee international conference on*, 155–164. IEEE. IEEE.

Antoulas, A. C., D. C. Sorensen, and S. Gugercin. 2001. A survey of model reduction methods for large-scale systems. *Contemporary Mathematics* 280: 193–219.

Bak, Stanley, Sergiy Bogomolov, and Taylor T. Johnson. 2015. HyST: A source transformation and translation tool for hybrid automaton models. In *18th international conference on hybrid systems: Computation and control*. Seattle, Washington: ACM.

Chahlaoui, Younes, and Paul Van Dooren. 2005. Benchmark examples for model reduction of linear time-invariant dynamical systems. In *Dimension reduction of large-scale systems*, 379–392. Springer.

De Moura, Leonardo, and Nikolaj Bjørner. 2008. Z3: An efficient smt solver. In *International conference on tools and algorithms for the construction and analysis of systems*, 337–340. Springer. Springer.

Duggirala, Parasara Sridhar, Sayan Mitra, and Mahesh Viswanathan. 2013. Verification of annotated models from executions. In *Proceedings of the eleventh acm international conference on embedded software. Emsoft '13*. Piscataway, NJ, USA: IEEE Press. ISBN 978-1-4799-1443-2.

Frehse, Goran, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. 2011. Spaceex: Scalable verification of hybrid systems. In *Computer aided verification*, 379–395. Springer. Springer.

Girard, Antoine, and George J Pappas. 2007. Approximate bisimulation relations for constrained linear systems. *Automatica* 43 (8): 1307–1317.

Girard, Antoine, A Agung Julius, and George J Pappas. 2008. Approximate simulation relations for hybrid systems. *Discrete Event Dynamic Systems* 18 (2): 163–179.

Girard, Antoine, George J Pappas, et al.. 2006. Approximate bisimulation for a class of stochastic hybrid systems. In *American control conference, 2006*, 6. IEEE. IEEE.

Han, Zhi. 2005. Formal verification of hybrid systems using model order reduction and decomposition. PhD diss, PhD thesis, Dept. of ECE, Carnegie Mellon University.

Han, Zhi, and Bruce Krogh. 2004. Reachability analysis of hybrid control systems using reduced-order models. In *American control conference, 2004. proceedings of the 2004*, Vol. 2, 1183–1189. IEEE. IEEE.

Islam, Md. Ariful, Abhishek Murthy, Ezio Bartocci, Elizabeth M. Cherry, Flavio H. Fenton, James Glimm, Scott A. Smolka, and Radu Grosu. 2015. Model-order reduction of ion channel dynamics using approximate bisimulation. *Theoretical Computer Science* 599: 34–46. doi:10.1016/j.tcs.2014.03.018. Advances in Computational Methods in Systems Biology.

Julius, A Agung. 2006. Approximate abstraction of stochastic hybrid automata. In *Hybrid systems: Computation and control*, 318–332. Springer.

Moore, Bruce. 1981. Principal component analysis in linear systems: Controllability, observability, and model reduction. *Automatic Control, IEEE Transactions on* 26 (1): 17–32.

Nesterov, Yurii, Arkadii Nemirovskii, and Yinyu Ye. 1994. *Interior-point polynomial algorithms in convex programming*, Vol. 13. SIAM.

Obinata, Goro, and Brian DO Anderson. 2012. *Model reduction for control system design.* Springer.

Pappas, George J. 2003. Bisimilar linear systems. *Automatica* 39 (12): 2035–2047.

Tabuada, Paulo, and George J Pappas. 2004. Bisimilar control affine systems. *Systems & Control Letters* 52 (1): 49–58.

Tanner, Herbert G, and George J Pappas. 2003. Abstractions of constrained linear systems. In *American control conference, 2003. proceedings of the 2003*, Vol. 4, 3381–3386. IEEE. IEEE.

van der Schaft, Arjan. 2004. Equivalence of dynamical systems by bisimulation. *IEEE transactions on automatic control* 49 (12): 2160–2172.

Vandenberghe, Lieven, and Stephen Boyd. 1994. Positive definite programming. *Mathematical Programming: State of the Art.*

7 Appendix

7.1 Proof of Theorem 1

Consider the uncontrolled augmented system $\dot{\bar{x}} = \bar{A}\bar{x}$, we have, $d(\bar{x}^T\bar{x})/dt = \bar{x}^T(\bar{A}+\bar{A}^T)\bar{x}$. It is easy to see that the controllability grammian and the observability grammian of the augmented system are the same (denoted as $\bar{\Sigma}$): $\bar{A}\bar{\Sigma} + \bar{\Sigma}\bar{A}^T + \bar{B}\bar{B}^T = 0$ and $\bar{A}^T\bar{\Sigma} + \bar{\Sigma}\bar{A} + \bar{C}^T\bar{C} = 0$. Combining two equations yields: $(\bar{A}+\bar{A}^T)\bar{\Sigma} + \bar{\Sigma}(\bar{A}+\bar{A}^T) = -\bar{B}\bar{B}^T - \bar{C}^T\bar{C}$. This Lyapunov equation implies that the real parts of all eigenvalues of $\bar{A}+\bar{A}^T$ are necessarily non-positive. Because $\bar{A}+\bar{A}^T$ is symmetric, its eigenvalues are real. Thus, these eigenvalues are either negative or zero. Note that $\bar{A}$ is asymptotically stable, hence $\bar{x}^T(t)\bar{x}(t) \to 0$ when $t \to \infty$ with all the initial state $\bar{x}(0)$. Combine all, we can conclude that $\bar{x}^T(t)\bar{x}(t)$ is monotonically converge to zero for any initial state $\bar{x}(0)$. Using the monotonic convergent property, the bound of the error e_1 satisfies: $\left\|e_1^i(t)\right\| = \left\|\bar{C}_i\bar{x}\right\| \le \left\|\bar{C}_i\right\|\|\bar{x}\| \le \left\|\bar{C}_i\right\|\|\bar{x}(0)\| \le \left\|\bar{C}_i\right\| \cdot \sup_{x(0)\in X_0}\|\bar{x}(0)\|$.

7.2 Proof of Theorem 2

Consider the uncontrolled augmented system (i.e. $u = 0$), let $V(\bar{x}(t)) = \bar{x}(t)^T P\bar{x}(t)$, we have $\dot{V}(\bar{x}(t)) = \bar{x}(t)^T(\bar{A}^T P + P\bar{A})\bar{x}(t)$. Assume P_0 is the solution of the optimization problem in Theorem 2. Because of $(\bar{A}^T P_0 + P_0\bar{A}) < 0$, then $V(\bar{x}(t)) < V(\bar{x}(0) = \bar{x}(0)^T P_0\bar{x}(0)$. Note that $\left\|e_1^i(t)\right\|^2 = \bar{x}^T\bar{C}_i^T\bar{C}_i\bar{x},\ 1 \le i \le p$. Since we also have $\bar{C}_i^T\bar{C}_i \le P_0$, the bound of the error satisfies $\left\|e_1^i(t)\right\| \le \sqrt{\bar{x}(0)^T P_0\bar{x}(0)}$.

7.3 Proof of Lemma 1

From the definition of output abstraction, we have:

$$\alpha_{ij}y_{r_j} - |\alpha_{ij}|\delta_j \le \alpha_{ij}y_j \le \alpha_{ij}y_{r_j} + |\alpha_{ij}|\delta_j \Rightarrow \Gamma y_r + \overline{\Psi}_2 \le \Gamma y + \Psi \le \Gamma y_r + \overline{\Psi}_1.$$

Thus, $S(M_k^\delta)$ and $U(M_k^\delta)$ defined by (3) satisfy the safety relation (1), which completes the proof.

7.3.1 Proof of Lemma 2

Let $\bar{y} = E(y-a)$, $\bar{y}_r = E(y_r - a)$. We have:

$$\begin{aligned}(y-a)^T Q(y-a) &= \bar{y}^T\Lambda\bar{y} = \sum\nolimits_{i=1}^p \lambda_i\bar{y}_i^2,\\ (y_r-a)^T Q(y_r-a) &= \bar{y}_r^T\Lambda\bar{y}_r = \sum\nolimits_{i=1}^p \lambda_i\bar{y}_{r_i}^2.\end{aligned} \tag{5}$$

From the definition of output abstraction (Definition 1), it is easy to see that:

$$-\bar{\delta}_i \le \bar{y}_i - \bar{y}_{r_i} = E(i,:)(y-y_r) \le \bar{\delta}_i,\ \bar{\delta}_i = \sum\nolimits_{j=1}^p |\gamma_{ij}|\delta_j. \tag{6}$$

Using (6) and the Cauchy-Schwarz inequality yields:

$$\begin{aligned}
&\sum\nolimits_{i=1}^{p} \lambda_i(\bar{y}_i - \bar{y}_{r_i})^2 \leq \varDelta_R^2 = \sum\nolimits_{i=1}^{p} \lambda_i \bar{\delta}_i^2, \\
&\sum\nolimits_{i=1}^{p} 2\lambda_i \bar{y}_{r_i}(\bar{y}_i - \bar{y}_{r_i}) \leq 2\varDelta_R \sqrt{\sum\nolimits_{i=1}^{p} \lambda_i \bar{y}_{r_i}^2}, \\
&\sum\nolimits_{i=1}^{p} 2\lambda_i \bar{y}_i(\bar{y}_{r_i} - \bar{y}_i) \leq 2\varDelta_R \sqrt{\sum\nolimits_{i=1}^{p} \lambda_i \bar{y}_i^2}.
\end{aligned} \tag{7}$$

From (7), the following inequalities are true:

$$\sum\nolimits_{i=1}^{p} \lambda_i \bar{y}_i^2 \leq (\sqrt{\sum\nolimits_{i=1}^{p} \lambda_i \bar{y}_{r_i}^2} + \varDelta_R)^2, \;\; \sum\nolimits_{i=1}^{p} \lambda_i \bar{y}_{r_i}^2 \leq (\sqrt{\sum\nolimits_{i=1}^{p} \lambda_i \bar{y}_i^2} + \varDelta_R)^2. \tag{8}$$

From (5) and (8), we have:

$$\begin{aligned}
\sqrt{(y-a)^T Q (y-a)} \leq \sqrt{(y_r - a)^T Q (y_r - a)} + \varDelta_R, \\
\sqrt{(y-a)^T Q (y-a)} \geq \sqrt{(y_r - a)^T Q (y_r - a)} - \varDelta_R.
\end{aligned} \tag{9}$$

Using (9), we can conclude that $S(M_k^\delta)$ and $S(M_k^\delta)$ defined in Lemma 2 satisfy the safety relation (1), which completes the proof.

Reachable Set Estimation and Control for Switched Linear Systems with Dwell-Time Restriction

Weiming Xiang, Hoang-Dung Tran and Taylor T. Johnson

***Abstract*— The reachable set estimation and control problems for continuous-time switched linear systems are addressed in this paper. First, a general result on reachable set estimation for switched system is proposed based on a Lyapunov function approach. Then, with the help of a class of time-scheduled Lyapunov functions, a numerically tractable sufficient condition ensuring the system state bounded in a prescribed set is derived for switched systems under dwell time constraint. Moreover, a time-scheduled state feedback controller is designed to ensure the state trajectories of the closed-loop system are confined in a prescribed set. Finally, a networked control system subject to packet dropouts is modeled as a switched system with dwell time constraints, and the controller design problem is studied as an application of our results.**

I. Introduction

Switched systems have emerged as an important class of hybrid systems and represent an active area of current research in the field of control systems [1]–[3]. A switched system is composed of a family of continuous or discrete-time subsystems along with a switching rule governing the switching between the subsystems. Generally, the stability and stabilization problems are the main concerns in the field of switched systems. It has been established that Lyapunov function techniques are effective to deal with stability and stabilization problems for switched systems, e.g. see [4]–[8]. Combining multiple Lyapunov function (MLF), the dwell time and average dwell time properties of relatively slowly switched systems have been investigated [9]–[15].

Reachable set estimation aims to derive a closed bounded set that constrains all the state trajectories generated by a dynamic system with a prescribed class of initial state set and inputs. Reachable set estimation is not only of theoretical interest in robust control theory [16], but also closely related to practical engineering for the safety verification problems [17]. In some early work, reachable set bounding was considered in the context of state estimation and it has later received a lot of attention in parameter estimation, see [18] and references therein. Recently, employing ellipsoidal techniques based on Lyapunov function approaches to estimate the reachable sets for different class of systems attracts many researchers' attention. In the framework of bounding ellipsoid, the quadratic Lyapunov function has played a fundamental role in the reachable set estimation problem, and it has been developed to time-delay systems [19]–[21], singular systems [22], discrete-time switched systems [23]. However, according to the best of the authors' knowledge, the reachable set estimation for continuous-time switched systems with constrained switching law, has not been fully investigated, and it motivates our study in this paper.

In this paper, the problems of reachable set estimation and control synthesis for continuous-time switched linear systems will be investigated. First, a general result based on Lyapunov function approach is presented. Then, under the framework of dwell time and with the help of a class of time-scheduled quadratic Lyapunov functions, a linear matrix inequality (LMI) based sufficient condition is proposed to estimate the reachable set. For the control synthesis, a time-scheduled feedback controller is designed to ensure the state trajectories being contained in a prescribed set and, moreover, an optimization problem is formulated to obtain an optimal controller gain to make the reachable set of closed-loop system as small as possible. As an application of our result, the control problem for a networked control system with package dropouts is studied. Based on our derived approach, the controller can be designed with an attempt to constrain state trajectories in a prescribed bounding ellipsoidal region.

Notation: The notations in this paper are fairly standard. $\mathbb{S}_{+}^{n\times n}$ is the set of real symmetric positive definite $n\times n$ matrices. In symmetric block matrices, we use * as an ellipsis for the terms that are introduced by symmetry. $\mathrm{diag}\{\cdots\}$ denotes a block-diagonal matrix and $\mathrm{int}[\cdot]$ rounds the element to the nearest integer towards zero.

II. Preliminaries and Problem Formulation

Let us consider a continuous-time switched linear system in the form of

$$\dot{x}(t) = A_{\sigma(t)}x(t) + B_{\omega,\sigma(t)}\omega(t) + B_{u,\sigma(t)}u(t) \qquad (1)$$

where $x(t)\in\mathbb{R}^{n_x}$ are the state of the system, and the initial state x_0 is assumed to be settled in a bounded ellipsoid as

$$x_0 \in \mathcal{X}_0 \triangleq \{x_0\in\mathbb{R}^{n_x} \mid x_0^\top R_0 x_0 \leq 1, R_0\in\mathbb{S}_{+}^{n_x\times n_x}\} \quad (2)$$

and $\omega(t)\in\mathbb{R}^{n_\omega}$ is the disturbance input vector which is assumed to satisfy the following ellipsoidal constraint

$$\omega(t) \in \mathcal{W} \triangleq \{\omega\in\mathbb{R}^{n_\omega} \mid \omega^\top R_\omega \omega \leq 1, R_\omega\in\mathbb{S}_{+}^{n_\omega\times n_\omega}\} \quad (3)$$

The material presented in this article is based on work sponsored by the Air Force Research Laboratory (AFRL) through contract number FA8750-15-1-0105 and the National Science Foundation (NSF) under grant number CNS-1464311. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of AFRL or NSF.

Authors are with the Department of Computer Science and Engineering, University of Texas at Arlington, Arlington, Texas 76019 USA. weiming.xiang@uta.edu; hoang-dung.tran@mavs.uta.edu; taylor.johnson@uta.edu

and $u(t) \in \mathbb{R}^{n_u}$ is the control input to be designed.

Define an index set $\mathcal{M} \triangleq \{1, 2, \ldots, N\}$, where N is the number of modes and, $\sigma : \mathbb{R}_{\geq 0} \to \mathcal{M}$ denotes the switching function, which is assumed to be a piecewise constant function continuous from right. The switching instants are expressed by a sequence $\mathcal{S} \triangleq \{t_k\}_{k\in\mathbb{N}}$, where t_0 denotes the initial time and t_k denotes the kth switching instant. Then, we define $\mathcal{I}_i \triangleq \{t \in \mathbb{R}_{\geq 0} \mid \sigma(t) = i, i \in \mathcal{M}\}$ to denote the activation time interval for ith mode.

The first problem considered in this paper is the reachable set estimation problem for switched system (1) with control input $u(t) = 0$, and the initial state satisfying (2), disturbance input satisfying (3). The reachable set is defined as

$$\mathcal{R}_x \triangleq \{x \in \mathbb{R}^{n_x} \mid x(t), x_0, \omega(t) \text{ satisfy } (1), (2), (3)\} \quad (4)$$

Then, the mode-dependent state feedback controller is considered, which has a time-scheduled structure as

$$u(t) = K_{\sigma(t)}(t)x(t) \quad (5)$$

Substituting above controller (5) into system (1), the closed-loop system becomes

$$\dot{x}(t) = \bar{A}_{\sigma(t)}(t)x(t) + B_{\omega,\sigma(t)}\omega(t) \quad (6)$$

where $\bar{A}_{\sigma(t)}(t) = A_{\sigma(t)} + B_{u,\sigma(t)}K_{\sigma(t)}(t)$.

The control objective is to ensure the state trajectory $x(t)$ contained in a given set

$$\tilde{\mathcal{R}}_x \triangleq \{x \in \mathbb{R}^{n_x} \mid x^\top R_x x \leq 1, R_x \in \mathbb{S}_+^{n_x \times n_x}\} \quad (7)$$

The above two problems are the main concerns in this paper. In the rest of this paper, the reachable set estimation problem will be studied at first, then based on the reachable set estimation results, the state feedback controller design problem will be addressed.

III. Reachable Set Estimation

A. General Lemma

First, a general lemma is presented to introduce the main idea to determine the over approximate set $\tilde{\mathcal{R}}_x$ for switched system (6), note that switched system (1) with $u(t) = 0$ is a particular case of $\bar{A}_i(t)$ being time-invariant.

Lemma 1: Consider switched system (6) under initial state condition (2) and disturbance input condition (3). If there exist a family of Lyapunov functions $V_i : \mathbb{R}^{n_x} \to \mathbb{R}_{\geq 0}$, $i \in \mathcal{M}$, satisfying $V_i(0) = 0$ and $V_i(x) > 0$, $\forall x \neq 0$, $\forall i \in \mathcal{M}$, and scalars $\alpha > 0$, $0 < \beta \leq 1$ such that

$$F_i(t) \leq 0, \forall t \in \mathcal{I}_i, \forall i \in \mathcal{M} \quad (8)$$

$$G_{i,j}(t_k) \leq 0, \forall t_k \in \mathcal{S}, i \neq j, \forall i, j \in \mathcal{M} \quad (9)$$

$$V_i(x_0) \leq x_0^\top R_0 x_0, \forall i \in \mathcal{M} \quad (10)$$

where $F_i(t) = \dot{V}_i(x(t)) + \alpha V_i(x(t)) - \alpha\omega^\top(t)R_\omega\omega(t)$ and $G_{i,j}(t_k) = V_i(x(t_k^+)) - \beta V_j(x(t_k^-)) + \beta - 1$. Then, the reachable set $\mathcal{R}_x$ satisfies $\mathcal{R}_x \subseteq \tilde{\mathcal{R}}_x \triangleq \{x \in \mathbb{R}^{n_x} \mid V_i(x) \leq 1, i \in \mathcal{M}\}$.

Proof: Define the following Lyapunov function as

$$V(t) = \sum\nolimits_{i\in\mathcal{M}} \xi_i(t)V_i(x(t)) \quad (11)$$

where $\xi_i : \mathbb{R}_{\geq 0} \to \{0, 1\}$ and $\sum_{i\in\mathcal{M}} \xi_i(t) = 1$ is the indicator function indicating the active modes at t.

First we consider any $t \in [t_k, t_{k+1}) \subset \mathcal{I}_i$, $\forall i \in \mathcal{M}$. (8) implies $\dot{V}(t) \leq -\alpha V(t) + \alpha\omega^\top(t)R_\omega\omega(t)$, $t \in [t_k, t_{k+1})$. Multiply both sides of this inequality with $e^{\alpha(t-t_k)}$ and then integrating it over $[t_k, t)$, we have $V(t) \leq e^{-\alpha(t-t_k)}V(t_k^+) + \int_{t_k}^t e^{-\alpha(t-s)}\omega^\top(s)R_\omega\omega(s)\mathrm{d}s$. Due to $\omega^\top(t)R_\omega\omega(t) \leq 1$, $\forall t \in \mathbb{R}_{\geq 0}$, we have the following result

$$\begin{aligned} V(t) &\leq e^{-\alpha(t-t_k)}V(t_k^+) + \int_{t_k}^t e^{-\alpha(t-s)}\mathrm{d}s \\ &= e^{-\alpha(t-t_k)}V(t_k^+) + 1 - e^{-\alpha(t-t_k)} \end{aligned} \quad (12)$$

and it can be rewritten to

$$V(t) - 1 \leq e^{-\alpha(t-t_k)}(V(t_k^+) - 1), \; t \in [t_k, t_{k+1}) \quad (13)$$

Next, we consider $t_k \in \mathcal{S}$. From (9), we can obtain $V(t_k^+) \leq \beta V(t_k^-) + 1 - \beta$, $t_k \in \mathcal{S}$, which equals to

$$V(t_k^+) - 1 \leq \beta(V(t_k^-) - 1), \; t_k \in \mathcal{S} \quad (14)$$

Combining (13) and (14), for $\forall t \in \mathbb{R}_{\geq 0}$, it can be obtained $V(t) - 1 \leq \cdots \leq \beta^{\mathrm{Num}(t-t_0)}e^{-\alpha(t-t_0)}(V(t_0) - 1)$, where $\mathrm{Num}(t - t_0)$ is the number of switchings in $[t_0, t)$. Due to $\alpha > 0$ and $0 < \beta \leq 1$, it means that $V(t) - 1 \leq V(t_0) - 1$, $\forall t \in \mathbb{R}_{\geq 0}$. Moreover, (10) implies $V(t_0) \leq x_0^\top R_0 x_0 \leq 1$, and it yields $V(t) \leq 1$, $\forall t \in \mathbb{R}_{\geq 0}$ holds, so $x(t) \in \tilde{\mathcal{R}}_x$, $\forall t \in \mathbb{R}_{\geq 0}$, where $\tilde{\mathcal{R}}_x \triangleq \{x \in \mathbb{R}^{n_x} \mid V_i(x) \leq 1, i \in \mathcal{M}\}$. ■

Although Lemma 1 provides a general framework to deal with the reachable set estimation problem, it is trivial in actual use, since it does not provide any available computational techniques for the construction of Lyapunov functions $V_i(x(t))$, $i \in \mathcal{M}$ and moreover, the proposed condition (9) requires us to check the values of Lyapunov functions at every the switching instant $t_k \in \mathcal{S}$. However, the switching instant sequence $\mathcal{S}$ usually cannot be specified in advance, and it is impossible to check Lemma 1 for all switching instants t_k in the case of $k \to \infty$.

B. Time-Scheduled Multiple Lyapunov Functions

Based on Lemma 1, we particularly consider a class of switched system with dwell-time constraint.

Definition 1: Given a switching signal function $\sigma(t)$ with a generated switching sequence $\mathcal{S}$, $\tau_{\min} = \inf_{k\in\mathbb{N}}\{t_{k+1} - t_k\}$ is called the minimum dwell time of $\sigma(t)$. $\mathcal{D}_{\tau_{\min}} \triangleq \{\sigma \mid \sigma : \mathbb{R}_{\geq 0} \to \mathcal{M}, t_{k+1} - t_k \geq \tau_{\min}, \forall k \in \mathbb{N}\}$ denotes the set of all switching policies with dwell time greater than $\tau_{\min}$.

Then, inspired by [11], [12], [15], we consider a class of time-scheduled multiple Lyapunov functions as follow

$$V_i(x(t)) = x^\top(t)P_i(t)x(t), \; t \in \mathbb{R}_{\geq 0}, \; i \in \mathcal{M} \quad (15)$$

where $P_i(t) \in \mathbb{S}_+^{n_x \times n_x}$, $i \in \mathcal{M}$ have the following structure:

Consider the interval $[t_k, t_k + \tau_{\min})$, we divide it into L segments described as $\mathcal{L}_{k,q} \triangleq [t_k + \theta_q, t_k + \theta_{q+1})$, $q = 0, 1, \ldots, L-1$ of equal lengths $h = \tau_{\min}/L$, and then $\theta_0 = 0$ and $\theta_q = qh = q\tau_{\min}/L$. We consider a class of continuous matrix function $P_i(t)$, $t \in [t_k, t_k + \tau_{\min})$ chosen to be linear within each segments $\mathcal{L}_{k,q}$, $q = 0, 1, \ldots, L - 1$. Explicitly, we can see that $\bigcup_{n=0}^{L-1}\mathcal{L}_{k,n} = [t_k, t_k + \tau_{\min})$ and $\mathcal{L}_{k,n} \cap \mathcal{L}_{k,m} = \emptyset$, $n \neq m$. Letting $P_{i,q} = P_i(t_k + \theta_q)$, then since the matrix function $P_i(t)$ is piecewise linear in $[t_k, t_k + \tau_{\min})$, it can be expressed in terms of the values at dividing points using a linear interpolation formula, that is, for $0 \leq \mu \leq 1$, $q = 0, 1, \ldots, L-1$,

$$P_i(t) = P_i(\mu) = (1-\mu)P_{i,q} + \mu P_{i,q+1},\ t \in \mathcal{L}_{k,q},\ i \in \mathcal{M} \tag{16}$$

where $\mu = L(t - t_k - \theta_q)/\tau_{\min}$.

As a result, the continuous matrix function $P_i(t) \in \mathbb{S}_{+}^{n_x \times n_x}$, $i \in \mathcal{M}$ can be completely determined by $P_{i,q} \in \mathbb{S}_{+}^{n_x \times n_x}$, $q = 0, 1, \ldots, L$, $i \in \mathcal{M}$, in interval $[t_k, t_k + \tau_{\min})$.

Then, due to $[t_k, t_k + \tau_{\min}) \subseteq [t_k, t_{k+1})$, for the remaining time in $[t_k, t_{k+1})$ denoted by $\mathcal{L}_{k,L} \triangleq [t_{k,\min}, t_{k+1})$, $P_i(t)$, $i \in \mathcal{M}$ is set to be

$$P_i(t) = P_{i,L},\ t \in \mathcal{L}_{k,L},\ i \in \mathcal{M} \tag{17}$$

In summary, the $P_i(t)$, $i \in \mathcal{M}$ in Lyapunov function in (15) is defined as

$$P_i(t) = \begin{cases} P_i(\mu), & t \in \mathcal{L}_{k,q},\ q = 0, 1, \ldots, L-1 \\ P_{i,L}, & t \in \mathcal{L}_{k,L} \end{cases} \tag{18}$$

where μ is defined in (16).

C. Reachable Set Estimation under Dwell Time Constraint

Now, we are ready to propose out main result as follows.

Theorem 1: Given dwell time $\tau_{\min} > 0$ and consider switched system (1) with $\sigma(t) \in \mathcal{D}_{\tau_{\min}}$ under initial state condition (2), disturbance input condition (3) and $u(t) = 0$. If there exist a set of matrices $P_{i,q} \in \mathbb{S}_{+}^{n_x \times n_x}$, $q = 0, 1, \ldots, L$, $i \in \mathcal{M}$ and a scalar $\alpha > 0$ such that for $\forall i, j \in \mathcal{M}$

$$\begin{bmatrix} \Xi_{i,q} + \Psi_{i,q} & * \\ B_{\omega,i}^{\top} P_{i,q} & -\alpha R_{\omega} \end{bmatrix} \prec 0,\ q = 0, \ldots, L-1 \tag{19}$$

$$\begin{bmatrix} \Xi_{i,q+1} + \Psi_{i,q} & * \\ B_{\omega,i}^{\top} P_{i,q} & -\alpha R_{\omega} \end{bmatrix} \prec 0,\ q = 0, \ldots, L-1 \tag{20}$$

$$\begin{bmatrix} \Xi_{i,L} & * \\ B_{\omega,i}^{\top} P_{i,L} & -\alpha R_{\omega} \end{bmatrix} \prec 0 \tag{21}$$

$$P_{i,0} - P_{j,L} \prec 0,\ i \neq j \tag{22}$$

$$P_{i,0} - R_0 \prec 0 \tag{23}$$

where $\Xi_{i,q} = A_i^{\top} P_{i,q} + P_{i,q} A_i + \alpha P_{i,q}$ and $\Psi_{i,g} = L(P_{i,q+1} - P_{i,q})/\tau_{\min}$. Then, the reachable set $\mathcal{R}_x \subseteq \tilde{\mathcal{R}}_x \triangleq \{x \in \mathbb{R}^{n_x} \mid x^{\top} P_{i,q} x \leq 1, q = 0, 1, \ldots, L, i \in \mathcal{M}\}$.

Proof: Construct Lyapunov function as

$$V(t) = \sum\nolimits_{i \in \mathcal{M}} \xi_i(t) x^{\top}(t) P_i(t) x(t) \tag{24}$$

where $P_i(t)$, $i \in \mathcal{M}$, is defined by (18) and $\xi_i(\cdot)$ is defined same as (11).

First, let us consider $F_i(t) = \dot{V}(t) + \alpha V(t) - \alpha \omega^{\top}(t) R_{\omega} \omega(t)$, which can be rewritten to

$$F_i(t) = \chi^{\top}(t) \begin{bmatrix} \Xi_i(t) + \dot{P}_i(t) & * \\ B_{\omega,i}^{\top} P_i(t) & -\alpha R_{\omega} \end{bmatrix} \chi(t) \tag{25}$$

where $\chi^{\top}(t) = [x^{\top}(t)\quad \omega^{\top}(t)]$ and $\Xi_i(t) = A_i^{\top} P_i(t) + P_i(t) A_i + \alpha P_i(t)$.

TABLE I

COMPUTATIONAL COMPLEXITIES OF THEOREM 1 WITH A FIXED α

Number of Decision Variables	LMI Constraints Size
$nN(L+1)(n+1)/2$	$2nN(N+2L+1)$

Suppose $\sigma(t) = i$, $t \in \mathcal{L}_{k,q}$, $q = 0, \ldots, L-1$, one has

$$\begin{bmatrix} \Xi_i(t) + \dot{P}_i(t) & * \\ B_{\omega,i}^{\top} P_i(t) & -\alpha R_{\omega} \end{bmatrix} = (1-\mu)\Pi_{i,1} + \mu \Pi_{i,2} \tag{26}$$

where $\Pi_{i,1} = \begin{bmatrix} \Xi_{i,q} + \Psi_{i,q} & * \\ B_{\omega,i}^{\top} P_{i,q} & -\alpha R_{\omega} \end{bmatrix}$ and $\Pi_{i,2} = \begin{bmatrix} \Xi_{i,q+1} + \Psi_{i,q+1} & * \\ B_{\omega,i}^{\top} P_{i,q+1} & -\alpha R_{\omega} \end{bmatrix}$.

Furthermore, we can see $\dot{P}_i(t) = (P_{i,q+1} - P_{i,q})\dot{\mu}$, $t \in \mathcal{L}_{k,q}$, $q = 0, \ldots, L-1$, and because of $\mu = L(t - t_k - \theta_q)/\tau_{\min}$, it implies $\dot{\mu} = L/\tau_{\min}$, leading to $\dot{P}_i(t) = \Psi_{i,q}$, $t \in \mathcal{L}_{k,q}$, $q = 0, \ldots, L-1$,. By (19), (20), it leads to

$$F_i(t) < 0,\ \forall t \in \bigcup\nolimits_{n=0}^{L-1} \mathcal{L}_{k,n} = [t_k, t_k + \tau_{\min}) \tag{27}$$

Then, we consider $t \in \mathcal{L}_{k,L}$. Since $P_i(t) = P_{i,L}, t \in \mathcal{L}_{k,L}$, we have $\dot{P}_i(t) = 0$, $\forall t \in \mathcal{L}_{k,L}$, thus (21) guarantees that $F_i(t) < 0$, $\forall t \in \mathcal{L}_{k,L}$. Together with (27), we can conclude that $F_i(t) < 0$, $\forall t \in \mathcal{I}_i$, $\forall i \in \mathcal{M}$, which means (8) in Lemma 1 holds.

Next, (22) ensures (9) holds with $\beta = 1$ and (23) guarantees (10) holds. Therefore, we have the reachable set $\mathcal{R}_x \subseteq \tilde{\mathcal{R}}_x \triangleq \{x \in \mathbb{R}^{n_x} \mid x^{\top} P_{i,q} x \leq 1, q = 0, 1, \ldots, L, i \in \mathcal{M}\}$ by Lemma 1. ∎

Remark 1: Parameter L implies the number of segments consisting of the dwell time interval $[t_k, t_k + \tau_{\min})$. A larger L yields a finer division of $[t_k, t_k + \tau_{\min})$, and a less conservative result can be consequently obtained, which will be demonstrated by a numerical example later. However, the computational cost increases as L grows, since a larger L inevitably introduces more decision variables and LMI constraints, see TABLE I for the computational complexity analysis for Theorem 1.

The set $\tilde{\mathcal{R}}_x$ is usually expected to be as small as possible to achieve a precise estimation of reachable set $\mathcal{R}_x$. Based on Theorem 2, one may add an additional constraint that

$$P_{i,q} \succeq \epsilon I,\ \epsilon > 0,\ \forall q = 0, 1, \ldots, L,\ \forall i \in \mathcal{M} \tag{28}$$

which implies that $\epsilon x^{\top}(t)x(t) \leq x^{\top}(t) P_{i,q} x(t) \leq 1$, namely $x(t) \in \mathcal{B}(0, 1/\sqrt{\epsilon}) \triangleq \{x \in \mathbb{R}^n \mid \|x\| \leq 1/\sqrt{\epsilon}\}$, $\forall t \in \mathbb{R}_{\geq 0}$, so we have to maximize ϵ to obtain a smallest reachable set with respect to ϵ. Given an L, the smallest ball $\mathcal{B}(0, 1/\sqrt{\epsilon}) \triangleq \{x \in \mathbb{R}^n \mid \|x\| < 1/\sqrt{\epsilon}\}$ containing the trajectories of state $x(t)$ in the framework of our approach can be obtained. Based on Theorem 1, an optimization problem can be formulated by adding (28) with (19)–(23) as follows

$$\max\ \epsilon \text{ s.t. } (28) \text{ and } (19) - (23) \tag{29}$$

In the extreme case with $L = 0$, $P_{i,q}$, shrinks to P_i, moreover, due to (35), we have to choose $P_i = P_j$, $i \neq j$. Thus, Theorem 2 is reduced to the following corollary.

Corollary 1: Consider switched system (1) under initial state condition (2), disturbance input condition (3) and $u(t) = 0$. If there exist a matrix $P \in \mathbb{S}_{+}^{n_x \times n_x}$ and a scalar $\alpha > 0$ such that

$$\begin{bmatrix} A_i^\top P + PA_i + \alpha P & * \\ B_{\omega,i}^\top P & -\alpha R_\omega \end{bmatrix} \prec 0, \ \forall i \in \mathcal{M} \tag{30}$$

$$P - R_0 \prec 0 \tag{31}$$

Then, the reachable set $\mathcal{R}_x \subseteq \tilde{\mathcal{R}}_x \triangleq \{x \in \mathbb{R}^{n_x} \mid x^\top P x \leq 1\}$.

Remark 2: Corollary 1 is actually the straightforward result derived based on the well-known common Lyapunov function approach. It can be observed that there is no restriction for the dwell time, this means that it can be used for arbitrary switching case which includes broader classes of switching signals, however, the cost is the increase of conservativeness of the estimation results.

IV. Time-Scheduled Feedback Controller Design

In this section, the controller design problem will be considered in the framework of dwell time. Based on Theorem 1, the following result can be derived for controller design.

Theorem 2: Given dwell time $\tau_{\min} > 0$ and consider switched system (1) with $\sigma(t) \in \mathcal{D}_{\tau_{\min}}$ under initial state condition (2) and disturbance $\omega(t)$ satisfying (3). If there exist a set of matrices $S_{i,q} \in \mathbb{S}_{+}^{n_x \times n_x}$, $X_{i,q} \in \mathbb{R}^{n_u \times n_x}$, $q = 0, 1, \ldots, L$, $i \in \mathcal{M}$ and a scalar $\alpha > 0$ such that for $\forall i, j \in \mathcal{M}$

$$\begin{bmatrix} \Xi_{i,q} - \Psi_{i,q} & * \\ B_{\omega,i}^\top & -\alpha R_\omega \end{bmatrix} \prec 0, \ q = 0, \ldots, L-1 \tag{32}$$

$$\begin{bmatrix} \Xi_{i,q+1} - \Psi_{i,q} & * \\ B_{\omega,i}^\top & -\alpha R_\omega \end{bmatrix} \prec 0, \ q = 0, \ldots, L-1 \tag{33}$$

$$\begin{bmatrix} \Xi_{i,L} & * \\ B_{\omega,i}^\top & -\alpha R_\omega \end{bmatrix} \prec 0 \tag{34}$$

$$S_{j,L} - S_{i,0} \prec 0, \ i \neq j \tag{35}$$

$$R_0^{-1} - S_{i,0} \prec 0 \tag{36}$$

$$S_{i,q} - R_x^{-1} \prec 0, \ q = 0, \ldots, L-1 \tag{37}$$

where $\Xi_{i,q} = A_i S_{i,q} + S_{i,q} A_i^\top + B_{i,u} X_{i,q} + X_{i,q}^\top B_{i,u}^\top + \alpha S_{i,q}$ and $\Psi_{i,q} = L(S_{i,q+1} - S_{i,q})/\tau_{\min}$. Then, the closed-loop system (6) with controller gain $K_i(t) = X_i(t) S_i^{-1}(t)$ has a reachable set $\mathcal{R}_x \subseteq \tilde{\mathcal{R}}_x \triangleq \{x \in \mathbb{R}^{n_x} \mid x^\top R_x x \leq 1\}$, where $S_i(t)$ and $X_i(t)$ are given by

$$S_i(t) = \begin{cases} (1-\mu) S_{i,q} + \mu S_{i,q+1} & t \in \mathcal{L}_{k,q} \\ S_{i,L} & t \in \mathcal{L}_{k,L} \end{cases} \tag{38}$$

$$X_i(t) = \begin{cases} (1-\mu) X_{i,q} + \mu X_{i,q} & t \in \mathcal{L}_{k,q} \\ X_{i,L} & t \in \mathcal{L}_{k,L} \end{cases} \tag{39}$$

where $\mu = L(t - t_k)/\tau_{\min} - q$ and q is determined by

$$q = \begin{cases} \mathrm{int}[L(t-t_k)/\tau_{\min}] & 0 \leq m < L \\ L & q \geq L \end{cases} \tag{40}$$

Proof: Since $S_{i,q} \succ 0$, it implies $S_i(t)$ defined by (60) is positive definite, and thus we have $S_i^{-1}(t) \succ 0$. Then, a Lyapunov function for closed-loop system (6) can be constructed as follows:

$$V(t) = \sum\nolimits_{i \in \mathcal{M}} \xi_i(t) x^\top(t) S_i^{-1}(t) x(t) \tag{41}$$

where $\xi_i(\cdot)$ is defined same as (11).

Substituting $X_i(t) = K_i(t) S_i(t)$, (32)–(34) ensure the following inequality holds

$$\begin{bmatrix} \bar{A}_i(t) S_i(t) + S_i(t) \bar{A}_i^\top(t) + \alpha S_i(t) - \dot{S}_i(t) & * \\ B_{\omega,i}^\top & -\alpha R_\omega \end{bmatrix} \prec 0 \tag{42}$$

Multiplying both side of (42) by $\mathrm{diag}\{S_i^{-1}(t), I\}$ and using $\dot{S}_i^{-1}(t) = -S_i^{-1}(t) \dot{S}_i(t) S_i^{-1}(t)$, we have

$$\begin{bmatrix} \Xi_i(t) & * \\ B_{\omega,i}^\top S_i^{-1}(t) & -\alpha R_\omega \end{bmatrix} \prec 0 \tag{43}$$

where $\Xi_i(t) = \bar{A}_i^\top(t) S_i^{-1}(t) + S_i^{-1}(t) \bar{A}_i(t) + \dot{S}_i^{-1}(t) + \alpha S_i^{-1}(t)$. It implies (8) in Lemma 1 holds.

Then, we consider (35) and (36). If (35) holds, it equals to $\Phi = \begin{bmatrix} -S_{j,L}^{-1} & I \\ I & -S_{i,0} \end{bmatrix} \prec 0$ by Schur complement. Then, further considering the Schur complement of Φ, we obtain $S_{i,0}^{-1} - S_{j,L}^{-1} \prec 0$ implying (9) in Lemma 1 holds with $\beta = 1$. Similarly, (10) can be guaranteed by (36). Thus, we have the reachable set $\mathcal{R}_x \subseteq \tilde{\mathcal{R}}_x \triangleq \{x \in \mathbb{R}^{n_x} \mid x^\top S_{i,q}^{-1} x \leq 1, q = 0, 1, \ldots, L, i \in \mathcal{M}\}$ by Lemma 1. Finally, from (37), we have $x^\top R_x x < x^\top S_{i,q}^{-1} x \leq 1$, $q = 0, 1, \ldots, L$, $i \in \mathcal{M}$, which implies $\mathcal{R}_x \subseteq \tilde{\mathcal{R}}_x \triangleq \{x \in \mathbb{R}^{n_x} \mid x^\top R_x x \leq 1\}$. ■

Remark 3: In order to obtain a optimized controller for the smallest reachable set estimation for closed loop system, we can add the following constraint

$$S_{i,q} - \delta I \prec 0, \ \delta > 0, q = 0, \ldots, L, i \in \mathcal{M} \tag{44}$$

The above inequality ensures the $x(t) \in \mathcal{B}(0, \sqrt{\delta}) \triangleq \{x \in \mathbb{R}^n \mid \|x\| \leq \sqrt{\delta}\}$. Given an L, the smallest ball $\mathcal{B}(0, \sqrt{\delta})$ containing the reachable set $\tilde{\mathcal{R}}_x$ can be obtained by the following optimization problem

$$\min \delta \text{ s.t. } (44) \text{ and } (32) - (36) \tag{45}$$

Corollary 2: Consider switched system (1) under initial state condition (2) and disturbance $\omega(t)$ satisfying (3). If there exist matrices $S \in \mathbb{S}_{+}^{n_x \times n_x}$, $X_i \in \mathbb{R}_{+}^{n_\omega \times n_x}$, $i \in \mathcal{M}$ and a scalar $\alpha > 0$ such that

$$\begin{bmatrix} \Xi_i & * \\ B_{\omega,i}^\top & -\alpha R_\omega \end{bmatrix} \prec 0, \ \forall i \in \mathcal{M} \tag{46}$$

$$R_0^{-1} \prec S \prec R_x^{-1} \tag{47}$$

where $\Xi_{i,q} = A_i S + S A_i^\top + B_{u,i} X_i + X_i^\top B_{u,i}^\top + \alpha S$. Then, the closed-loop system (6) with controller gain $K_i = X_i S$ has a reachable set $\mathcal{R}_x \subseteq \tilde{\mathcal{R}}_x \triangleq \{x \in \mathbb{R}^{n_x} \mid x^\top R_x x \leq 1\}$.

Proof: It can be easily proved by letting $L = 0$ in Theorem 2, so the proof is omitted here. ■

Though Corollary 2 provides constant feedback gains K_i, $i \in \mathcal{M}$ which does not require online computation as $K_i(t)$, $i \in \mathcal{M}$ do. This feature is more convenient for controller realization in practice, but the conservatism grows in comparison with the case of $L > 0$.

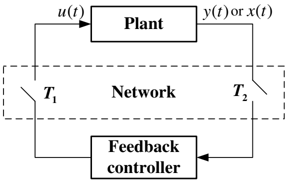


Fig. 1. Packet dropouts in networked control system

V. Application in Networked Control Systems

Consider a networked control system with packet dropouts in both forward and backward channels, where the packet dropouts can be modeled as switches open behavior, which is illustrated in Fig. 1. When T_1 is closed, the controller output is successfully transmitted to the actuator; whereas when it is open, the output of the switch becomes zero and a packet is lost, and we have in this case $u(t) = 0$. The situation is the same for the backward channel. In absence of packet dropouts, the state feedback controller works well during interval $\Gamma_{1,k} \triangleq [t_{2k}, t_{2k+1}), k \in \mathbb{N}$. However, due to the occurrence of packet dropouts, the controller is considered to be not available, namely $u(t) = 0$, in the time interval $\Gamma_{2,k} \triangleq [t_{2k+1}, t_{2k+2}), k \in \mathbb{N}$.

Assumption 1: The following assumptions are made:

1) There exists a uniform lower-bound $\tau_{\min}$ on the lengths of $\Gamma_{1,k}$, $k \in \mathbb{N}$, that is $t_{2k+1} - t_{2k} \geq \tau_{\min}$, $\forall k \in \mathbb{N}$.
2) There exist a uniform upper-bound $\psi_{\max}$ on the lengths of $\Gamma_{2,k}$, $k \in \mathbb{N}$, that is $t_{2k+2} - t_{2k+1} \leq \psi_{\max}$, $\forall k \in \mathbb{N}$.

The plant we consider is a linear system

$$\dot{x}(t) = Ax(t) + B_\omega \omega(t) + B_u u(t) \tag{48}$$

and the controller is considered to be $u(t) = K(t)x(t),\ t \in \Gamma_{1,k}$. In summary, the networked control system with packet dropouts can be described as follows

$$\dot{x}(t) = A_{\sigma(t)}(t)x(t) + B_\omega \omega(t) \tag{49}$$

where $A_1(t) = A + B_u K(t)$ and $A_2(t) = A$, and the switching function $\sigma(t)$ is

$$\sigma(t) = \begin{cases} 1 & t \in \Gamma_{1,k} \\ 2 & t \in \Gamma_{2,k} \end{cases} \tag{50}$$

Theorem 3: Under Assumption 1 and consider networked control system (48) under initial state condition (2) and disturbance $\omega(t)$ satisfying (3). If there exist a set of matrices $S_{i,q} \in \mathbb{S}_+^{n_x \times n_x}$, $X_{i,q} \in \mathbb{R}_+^{n_u \times n_x}$, $q = 0, 1, \ldots, L$, $i = 1, 2$ and a scalar $\alpha > 0$ such that

$$\begin{bmatrix} \Xi_{1,q} - \Psi_{1,q} & * \\ B_\omega^\top & -\alpha R_\omega \end{bmatrix} \prec 0,\ q = 0, \ldots, L-1 \tag{51}$$

$$\begin{bmatrix} \Xi_{1,q+1} - \Psi_{1,q} & * \\ B_\omega^\top & -\alpha R_\omega \end{bmatrix} \prec 0,\ q = 0, \ldots, L-1 \tag{52}$$

$$\begin{bmatrix} \Xi_{1,L} & * \\ B_\omega^\top & -\alpha R_\omega \end{bmatrix} \prec 0 \tag{53}$$

$$\begin{bmatrix} \Xi_{2,q} - \Psi_{2,q} & * \\ B_\omega^\top & -\alpha R_\omega \end{bmatrix} \prec 0,\ q = 0, \ldots, L-1 \tag{54}$$

$$\begin{bmatrix} \Xi_{2,q+1} - \Psi_{2,q} & * \\ B_\omega^\top & -\alpha R_\omega \end{bmatrix} \prec 0,\ q = 0, \ldots, L-1 \tag{55}$$

$$S_{1,L} - S_{2,0} \prec 0 \tag{56}$$

$$S_{2,q} - S_{1,0} \prec 0,\ q = 0, \ldots, L \tag{57}$$

$$R_0^{-1} - S_{i,0} \prec 0,\ i = 1, 2 \tag{58}$$

$$S_{i,q} - R_x^{-1} \prec 0,\ i = 1, 2,\ q = 0, \ldots, L \tag{59}$$

where $\Xi_{1,q} = AS_{1,q} + S_{1,q}A^\top + B_u X_{1,q} + X_{1,q}^\top B_u^\top + \alpha S_{1,q}$, $\Xi_{2,q} = AS_{i,q} + S_{2,q}A^\top + \alpha S_{2,q}$ and $\Psi_{1,q} = L(S_{1,q+1} - S_{1,q})/\tau_{\min}$, $\Psi_{2,q} = L(S_{2,q+1} - S_{2,q})/\psi_{\max}$. Then, the closed-loop system (49) with controller gain $K(t) = X_1(t)S_1^{-1}(t)$, $t \in \Gamma_{1,k}$ has a reachable set $\mathcal{R}_x \subseteq \tilde{\mathcal{R}}_x \triangleq \{x \in \mathbb{R}^{n_x} \mid x^\top R_x x \leq 1\}$, where $S_1(t)$ and $X_1(t)$ are

$$S_1(t) = \begin{cases} (1-\mu)S_{1,q} + \mu S_{1,q+1} & t \in \mathcal{L}_{k,q} \\ S_{1,L} & t \in \mathcal{L}_{k,L} \end{cases} \tag{60}$$

$$X_1(t) = \begin{cases} (1-\mu)X_{1,q} + \mu X_{1,q} & t \in \mathcal{L}_{k,q} \\ X_{1,L} & t \in \mathcal{L}_{k,L} \end{cases} \tag{61}$$

where $\mu = L(t - t_{2k})/\tau_{\min} - q$ and q is determined by

$$q = \begin{cases} \mathrm{int}[L(t - t_{2k})/\tau_{\min}] & 0 \leq m < L \\ L & q \geq L \end{cases} \tag{62}$$

Proof: By the similar guidelines in Theorem 2, conditions (51), (52) and (53) ensures that (8) in Lemma 1 holds for interval $\Gamma_{1,k}$, and (54), (55) guarantee (8) in Lemma 1 holds for $\Gamma_{2,k}$. Then, (56) and (57) implies (9) holds for switching instants t_{2k+1}, t_{2k}, respectively. Finally, (10) can be guaranteed by (58). Thus, according Lemma 1, the reachable set is obtained as $\mathcal{R}_x \subseteq \tilde{\mathcal{R}}_x \triangleq \{x \in \mathbb{R}^{n_x} \mid x^\top S_{i,q}^{-1} x \leq 1, q = 0, 1, \ldots, L, i = 1, 2\}$. Using (59), we have $\mathcal{R}_x \subseteq \tilde{\mathcal{R}}_x \triangleq \{x \in \mathbb{R}^{n_x} \mid x^\top R_x x \leq 1\}$. ■

By adding the following constraint

$$S_{i,q} - \delta I \prec 0,\ \delta > 0,\ q = 0, \ldots, L,\ i = 1, 2 \tag{63}$$

The smallest ball $\mathcal{B}(0, \sqrt{\delta})$ containing the reachable set $\tilde{\mathcal{R}}_x$ can be obtained by the following optimization problem

$$\min \delta \text{ s.t. } (63) \text{ and } (51) - (58) \tag{64}$$

Example 1: Consider the plant described by

$$A = \begin{bmatrix} 1.5 & 2.5 \\ 1.5 & 1.2 \end{bmatrix},\ B_u = \begin{bmatrix} 0.2 \\ 0.5 \end{bmatrix},\ B_\omega = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

The initial state is assumed to satisfy $x_0 \in \{x_0 \in \mathbb{R}^n \mid \|x_0\| \leq 1\}$, and the control objective is to ensure the state trajectories satisfies $x(t) \in \{x \in \mathbb{R}^n \mid \|x\| \leq 2\}$. Assume that the minimal reliable time for a group of successfully transmitted information is $\tau_{\min} = 0.5$ second, and the maximal time for a group of successive packet dropouts is $\psi_{\max} = 0.1$ second. Let $\alpha = 0$ due to $B_\omega = [0 \ \ 0]^\top$, and we can find feasible solution to LMIs (51)–(59) with $L = 1$.

Given an initial state $x_0 = [0.6 \ \ 0.8]^\top$, the state response is illustrated in Fig. 2, it can be observed that the state trajectory satisfies $x(t) \in \{x \in \mathbb{R}^n \mid \|x\| \leq 2\}$. Moreover, we generate 500 random state trajectories with random packet dropouts whose lengths are less than 0.1 second, it can be seen that all the trajectories are in the prescribed ball $\mathcal{B}(0, 2)$, which are shown in Fig. 3.

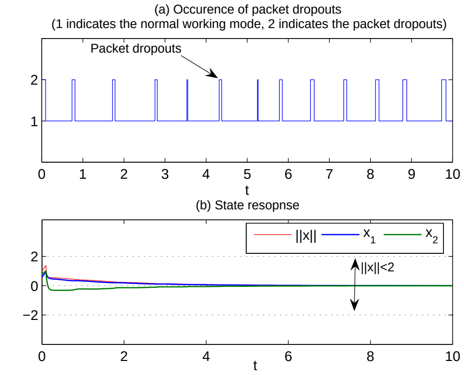


Fig. 2. State response of networked control system with packet dropouts

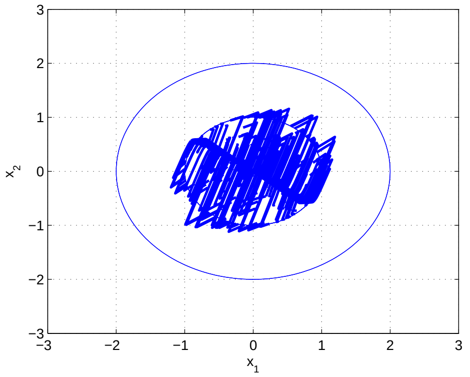


Fig. 3. 500 random state trajectories with random packet dropouts. All the trajectories $x(t)$ generated from the $\|x_0\| \leq 1$ are bounded by $\|x(t)\| \leq 2$.

Finally, in order the show how parameter L works for the controller design, different L are selected for optimization problem (64). From $L = 1$ to $L = 5$, the smallest δ are computed, which are shown in Table II. In Table II, we can see that δ monotonically decreases as L increases, this is consistent with Remark 1. However, a selection of larger L has to afford more computational cost, the computation time grows as L increases in Table II.

TABLE II

δ AND COMPUTATION TIME (C.T.) WITH $L = 1, 2, 3, 4, 5$

	$L = 1$	$L = 2$	$L = 3$	$L = 4$	$L = 5$
δ	1.8795	1.5075	1.4615	1.4434	1.4334
C.T.	0.296 s	0.433 s	0.561 s	0.734 s	0.905 s

VI. Conclusions

By employing a class of time-scheduled Lyapunov functions, the reachable estimation and control problems for switched linear systems under dwell time constraint are investigated in this paper. A sufficient condition has been proposed to estimate the reachable set of switched system by bounding ellipsoids, then based on the estimation result, a time-scheduled state feedback controller gains are obtained, which can ensure the state trajectories of closed-loop system in a prescribed set. Finally, the controller design result is applied into the networked control system with packet dropouts.

References

[1] D. Liberzon, *Switching in systems and control*. Springer Science & Business Media, 2012.

[2] R. A. DeCarlo, M. S. Branicky, S. Pettersson, and B. Lennartson, "Perspectives and results on the stability and stabilizability of hybrid systems," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 1069–1082, 2000.

[3] M. S. Mahmoud, *Switched time-delay systems*. Springer, 2010.

[4] H. Lin and P. J. Antsaklis, "Stability and stabilizability of switched linear systems: a survey of recent results," *IEEE Transactions on Automatic Control*, vol. 54, no. 2, pp. 308–322, 2009.

[5] R. Shorten, F. Wirth, O. Mason, K. Wulff, and C. King, "Stability criteria for switched and hybrid systems," *SIAM review*, vol. 49, no. 4, pp. 545–592, 2007.

[6] M. Margaliot, "Stability analysis of switched systems using variational principles: an introduction," *Automatica*, vol. 42, no. 12, pp. 2059–2077, 2006.

[7] M. S. Branicky, "Multiple lyapunov functions and other analysis tools for switched and hybrid systems," *IEEE Transactions on Automatic Control*, vol. 43, no. 4, pp. 475–482, 1998.

[8] J. Daafouz, P. Riedinger, and C. Iung, "Stability analysis and control synthesis for switched systems: a switched lyapunov function approach," *IEEE Transactions on Automatic Control*, vol. 47, no. 11, pp. 1883–1887, 2002.

[9] A. S. Morse, "Supervisory control of families of linear set-point controllers part i. exact matching," *IEEE Transactions on Automatic Control*, vol. 41, no. 10, pp. 1413–1431, 1996.

[10] J. P. Hespanha and A. S. Morse, "Stability of switched systems with average dwell-time," in *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, vol. 3, pp. 2655–2660, IEEE, 1999.

[11] L. Allerhand and U. Shaked, "Robust stability and stabilization of linear switched systems with dwell time," *IEEE Transactions on Automatic Control*, vol. 56, no. 2, pp. 381–386, 2011.

[12] C. Briat and A. Seuret, "A looped-functional approach for robust stability analysis of linear impulsive systems," *Systems & Control Letters*, vol. 61, no. 10, pp. 980–988, 2012.

[13] L. Zhang, S. Zhuang, and P. Shi, "Non-weighted quasi-time-dependent h filtering for switched linear systems with persistent dwell-time," *Automatica*, vol. 54, pp. 201–209, 2015.

[14] W. Xiang, "On equivalence of two stability criteria for continuous-time switched systems with dwell time constraint," *Automatica*, vol. 54, pp. 36–40, 2015.

[15] W. Xiang and J. Xiao, "Stabilization of switched continuous-time systems with all modes unstable via dwell time switching," *Automatica*, vol. 50, no. 3, pp. 940–945, 2014.

[16] E. Fridman, A. Pila, and U. Shaked, "Regional stabilization and $\mathcal{H}_\infty$ control of time-delay systems with saturating actuators," *International Journal of Robust and Nonlinear Control*, vol. 13, no. 9, pp. 885–907, 2003.

[17] J. Lygeros, C. Tomlin, and S. Sastry, "Controllers for reachability specifications for hybrid systems," *Automatica*, vol. 35, no. 3, pp. 349–370, 1999.

[18] C. Durieu, E. Walter, and B. Polyak, "Multi-input multi-output ellipsoidal state bounding," *Journal of optimization theory and applications*, vol. 111, no. 2, pp. 273–303, 2001.

[19] E. Fridman and U. Shaked, "On reachable sets for linear systems with delay and bounded peak inputs," *Automatica*, vol. 39, no. 11, pp. 2005–2010, 2003.

[20] Z. Feng and J. Lam, "An improved result on reachable set estimation and synthesis of time-delay systems," *Applied Mathematics and Computation*, vol. 249, pp. 89–97, 2014.

[21] B. Zhang, J. Lam, and S. Xu, "Reachable set estimation and controller design for distributed delay systems with bounded disturbances," *Journal of the Franklin Institute*, vol. 351, no. 6, pp. 3068–3088, 2014.

[22] Z. Feng and J. Lam, "On reachable set estimation of singular systems," *Automatica*, vol. 52, pp. 146–153, 2015.

[23] Y. Chen, J. Lam, and B. Zhang, "Estimation and synthesis of reachable set for switched linear systems," *Automatica*, vol. 63, pp. 122–132, 2016.

On Reachable Set Estimation for Discrete-Time Switched Linear Systems under Arbitrary Switching

Weiming Xiang, Hoang-Dung Tran and Taylor T. Johnson

Abstract— **This paper addresses the problem of reachable set estimation for discrete-time switched systems under arbitrary switching. By introducing a novel conception called M-step sequence which is capable of characterizing all possible subsystem activation orders during M discrete-time steps, a Lyapunov function based approach is proposed to derive a set of bounding ellipsoids to estimate the reachable set. The proposed approach can cover the previous switched Lyapunov function approach and yields less conservativeness. Moreover, it can be shown that the M-step sequence method can also reduce the conservativeness in stability analysis for discrete-time switched systems under arbitrary switching in contrast to switched Lyapunov function method. Several numerical examples are provided to illustrate our approach.**

I. Introduction

A switched system is composed of a family of continuous or discrete-time subsystems, described by differential or difference equations, respectively, along with a switching rule governing the switching between the subsystems. The motivation for studying such switched systems comes from the fact that switched system can be efficiently used to model many practical systems that are inherently multi-model, thus several dynamical subsystem models are required to describe their behavior. For example, several real-world cyber-physical systems and industrial processes exhibit switching and hybrid nature intrinsically. Generally, the stability and stabilization problems are the main concerns in the field of switched systems, e.g., see [1]–[4] and the references cited therein. One can study the stability and other properties of switched systems with a given the switching rule as a prescribed state space partitioning [5]–[7] or with some known constraints on switching sequence such as dwell time [8] or average dwell time [9] restrictions. For instance, combining multiple Lyapunov function (MLF), the dwell time and average dwell time properties of relatively slowly switched systems have been investigated in the corresponding switched systems [10]–[16]. However, in a number of practical switched systems, the switching sequence is not known *a prior* and these properties have to be examined under arbitrary switching.

Reachable set estimation aims to derive a closed bounded set that contains all the state trajectories generated by a dynamic system with a prescribed class of initial state set and inputs. Reachable set estimation is not only of theoretical interest in robust control theory [17], but also closely related to practical engineering for the safety verification problems [18]. For example, a dynamic system is regarded to be safe if its reachable set does not intersect with the unsafe or undesirable sets of states. In some early work, reachable set bounding was considered in the context of state estimation and it has later received a lot of attention in parameter estimation, see [19] and references therein. Recently, employing ellipsoidal techniques based on Lyapunov function approaches to estimate the reachable sets for different class of systems attracts many researchers' attention. In the framework of bounding ellipsoid, the quadratic Lyapunov function has played a fundamental role in the reachable set estimation problem, and it has been further extended and developed to time-delay systems [20]–[22], singular systems [23].

For the reachable set estimation problem for discrete-time switched system under arbitrary switching, [24] proposes a method based on switched Lyapunov function approach, and the trajectories are estimated by a set of bounding ellipsoids. The main aim in this paper is to further develop the Lyapunov function approach and reduce its conservativeness in reachable set estimation for discrete-time switched system under arbitrary switching. By introducing the conception of M-step sequence which is able to characterize all possible subsystem activation orders during M steps, an improved method will be proposed in this paper. It should be stressed that the approach in [24] can be recovered by particularly letting $M = 1$ and thus has less conservativeness. Additionally, also in virtue of the advantages of M-step sequence, the less conservativeness emerges in stability analysis for discrete-time switched system in comparison with the well-known switched Lyapuonv function. Finally, several numerical examples are given in order to emphasize the less conservativeness and effectiveness of the approach.

The remainder of this paper is organized as follows: Preliminaries and problem formulation are given in Section II. The main results including the M-step sequence, reachable set estimation and discussion on stability are given in Section III. Numerical examples are provided in Section IV. Conclusions are given in Section V.

Notation: $\mathbb{N}$ represents the set of natural numbers. $\mathbb{R}$ and $\mathbb{R}_{\geq 0}$ denote the fields of real numbers and nonnegative real numbers, respectively. $\mathbb{R}^n$ is the vector space of all n-tuples of real numbers, $\mathbb{R}^{n \times n}$ is the space of $n \times n$ matrices with

The material presented in this paper is based upon work supported by the National Science Foundation (NSF) under grant numbers CNS 1464311, EPCN 1509804, and SHF 1527398, the Air Force Research Laboratory (AFRL) through contract number FA8750-15-1-0105, and the Air Force Office of Scientific Research (AFOSR) under contract numbers FA9550-15-1-0258 and FA9550-16-1-0246.

Authors are with the Department of Electrical Engineering and Computer Science, Vanderbilt University, Nashville, Tennessee 37212, USA. xiangwming@gmail.com; trhoangdung@gmail.com; taylor.johnson@gmail.com

real entries. The notation $P \succ 0$ ($P \prec 0$) means P is real symmetric and positive definite (negative definite). $A^\top$ denotes the transpose of A. In symmetric block matrices, we use $*$ as an ellipsis for the terms that are introduced by symmetry. $\mathrm{diag}\{\cdots\}$ denotes a block-diagonal matrix. $|\cdot|$ stands for the Euclidean norm. The bounding ellipsoid is expressed by $\mathcal{E}(R) \triangleq \{x \in \mathbb{R}^n \mid x^\top Rx \leq 1, 0 \prec R \in \mathbb{R}^{n\times n}\}$, and ball $\mathcal{B}(x_0, \delta) \triangleq \{x \in \mathbb{R}^n \mid |x - x_0| \leq \delta, x_0 \in \mathbb{R}^n, \delta > 0\}$.

II. Preliminaries and Problem Formulation

In this paper, we consider a class of discrete-time switched linear system in the following form

$$x(k+1) = A_{\sigma(k)}x(k) + B_{\sigma(k)}\omega(k) \qquad (1)$$

where $x(k), x_0 \in \mathbb{R}^{n_x}$ are the state of the system and initial state, respectively. The switching signal σ is defined as $\sigma : \mathbb{N} \to \mathcal{I}[1, N]$, where N is the number of subsystems involved in the switched system. In this paper, no specific restriction is imposed on switching signal σ, namely the arbitrary switching law is considered in the rest of paper. A_i, and B_i, $i \in \mathcal{I}[1, N]$ are constant system matrices with appropriate dimensions. $\omega(k) \in \mathbb{R}^{n_\omega}$ is the bounded peak input vector which is assumed to satisfy the following constraint

$$\omega(k) \in \mathcal{W} \triangleq \{\omega \in \mathbb{R}^{n_\omega} \mid \omega^\top\omega \leq d^2, d > 0\} \qquad (2)$$

The main problem considered in this paper is the reachable set estimation problem for switched system (1) with input $\omega(k)$ satisfying (2). The reachable set is defined as

$$\mathcal{R}_x \triangleq \{x \in \mathbb{R}^{n_x} \mid x_0 = 0, x(k), \omega(k) \text{ satisfy (1), (2)}\} \quad (3)$$

Due to the complex characteristic of switched systems, the accurate reachable set for switched system (1) is hard to compute. The reachable set estimation problem is formulated as follows.

Problem 1: For switched system (1), determine an over approximate set $\tilde{\mathcal{R}}_x$ such that $\mathcal{R}_x \subseteq \tilde{\mathcal{R}}_x$, and the set $\tilde{\mathcal{R}}_x$ should be optimized as small as possible.

The recent solution to compute an over approximate set $\tilde{\mathcal{R}}_x$ is proposed in [24], which is based on switched Lyapunov function approach [25].

Lemma 1: [24] Consider system (1) with input (2). If there exist a set of a family of functions $V_i : \mathbb{R}^n \to \mathbb{R}_+$ satisfying $V_i(0) = 0$ and $V_i(x) > 0$, $\forall x = 0$, $i \in \mathcal{I}[1, N]$, and exist scalars $0 < \alpha_{i,j} < 1$ such that $\forall(i, j) \in \mathcal{I}[1, N] \times \mathcal{I}[1, N]$,

$$V_j(x(k+1)) - \alpha_{i,j}V_i(x(k)) - \frac{1-\alpha_{i,j}}{d^2}\omega^\top(k)\omega(k) \leq 0 \qquad (4)$$

then system (1) is globally uniformly asymptotically stable and we have $\exists i \in \mathcal{I}[1, N]$ such that $V_i(x(k)) \leq 1$ for all $x(0)$ satisfying $V_i(x(0)) \leq 1$, $\forall i \in \mathcal{I}[1, N]$.

In the framework of quadratic switched Lyapunov function, the following result for reachable set estimation stems from above Lemma.

Lemma 2: [24] Consider system (1) with input (2). If there exist matrices $P_i \succ 0$, $i \in \mathcal{I}[1, N]$ and scalars $0 < \alpha_{i,j} < 1$ such that $\forall(i, j) \in \mathcal{I}[1, N] \times \mathcal{I}[1, N]$,

$$\begin{bmatrix} A_i^\top P_jA_i - \alpha_{i,j}P_i & A_i^\top P_jB_i \\ * & B_i^\top P_jB_i - \frac{1-\alpha_{i,j}}{d^2}I \end{bmatrix} \preceq 0 \qquad (5)$$

then system (1) is GUAS and the reachable set $\mathcal{R}_x$ can be over approximated by $\tilde{\mathcal{R}}_x \triangleq \bigcup_{i\in\mathcal{I}[1,N]} \mathcal{E}(P_i)$.

Remark 1: In [24], Lemma 1 has $V_i(x(k)) \leq 1$, $\forall i \in \mathcal{I}[1, N]$, and the reachable set $\mathcal{R}_x$ in Lemma 2 is bounded by the intersection of a set of ellipsoids $\bigcap_{i\in\mathcal{I}[1,N]} \mathcal{E}(P_i)$. We correct this slight error as that $\exists i \in \mathcal{I}[1, N]$ such that $V_i(x(k)) \leq 1$ and the over approximate set $\tilde{\mathcal{R}}_x$ should be the union of a set of ellipsoids $\bigcup_{i\in\mathcal{I}[1,N]} \mathcal{E}(P_i)$, since $\sigma(k)$ is an arbitrary switching means $\sigma(k)$ could be any possible $i \in \mathcal{I}[1, N]$ and $\tilde{\mathcal{R}}_x$ needs to include all possible trajectories for any $i \in \mathcal{I}[1, N]$.

On the basis of above lemma, the over approximate reachable set $\tilde{\mathcal{R}}_x$ can be characterized by a set of ellipsoids, and optimization problems can be formulated to obtain $\tilde{\mathcal{R}}_x$ as small as possible in [24]. In this paper, our main aim is to further improve this Lyapunov function based approach to develop less conservative result for reachable set estimation of switched system (1), namely to develop an approach to better over approximate the reachable set $\mathcal{R}_x$.

III. Main Results

In this section, the reachable set estimation problem will be studied based on a novel conception named M-step sequence, an LMI based approach will be proposed to obtain a set of bounding ellipsoids. Moreover, the globally uniformly asymptotical stability of discrete-time switched linear system is discussed in the framework of M-step sequence.

A. M-Step Sequence

In this paper, the main aim is to further reduce the conservatism in Lyapunov function based approach for reachable set estimation of discrete-time switched system over switched Lyapunov function methods. First, we introduce the conception of the M-step sequence, which plays a fundamental role in this paper. The M-step sequence is defined as follows.

Definition 1: For a switched system consisting N subsystem, and given a time window with M-step length, an M-step sequence is a combination of subsystems in M steps. There are N^M combinations of subsystems in M steps, and these N^M combinations are indexed by $\mathcal{I}[1, N^M]$. For the ith sequence of combination in $\mathcal{I}[1, N^M]$, it is expressed by

$$\mathcal{S}_i^M \triangleq \{i_1, i_2, \ldots, i_M\},\ i_1, \ldots, i_M \in \mathcal{I}[1, N],\ i \in \mathcal{I}[1, N^M]$$

The M-step sequence is able to characterize all possible activation orders for switched system during the M steps. Given a switching signal $\sigma(k)$ in $[0, \infty)$, we denote the nth M-step activation sequence is

$$\Sigma_n \triangleq \{\sigma(nM), \sigma(nM+1), \ldots, \sigma((n+1)M-1)\} \qquad (6)$$

where $n = 0, 1, \ldots$.

The following properties can be easily observed.

Proposition 1: Given any switching signal $\sigma(k)$ defined over interval $[0, \infty)$, we have

1) $\bigcup_{n=0}^{\infty} \Sigma_n = \{\sigma(0), \sigma(1), \sigma(2), \ldots\}$.
2) For any $n = 0, 1, \ldots$, there exists an $i \in \mathcal{I}[1, N^M]$ such that $\Sigma_n = \mathcal{S}_i^M$.

Remark 2: The first property implies the activation order of any switching signal $\sigma(k)$ can be expressed by M-step activation sequence Σ_n as $n \to \infty$. The second property means any M-step activation sequence Σ_n can be found in $\mathcal{S}_i^M$, $i \in \mathcal{I}[1, N^M]$. These two properties suffice to show that the M-step sequence $\mathcal{S}_i^M$, $i \in \mathcal{I}[1, N^M]$, is capable to describe the behaviors of switching signal $\sigma(k)$ in $[0, \infty)$.

Based on the introduced notion of M-step sequence, we introduce a class of M-step clock-dependent switched Lyapunov functions $V_i : [nM + 1, (n + 1)M] \times \mathbb{R}^n \to \mathbb{R}_+$, $n = 0, 1, \ldots$, $i \in \mathcal{I}[1, N^M]$, associated to M-step sequence, which are a family of non-negative functions satisfying

$$\beta_1(|x|) < V_i(k, x) < \beta_2(|x|) \tag{7}$$

where $\beta_1, \beta_2 \in \mathcal{K}_\infty$.

In the framework of the M-step clock-dependent switched Lyapunov function, the following result can be obtained as an improvement of Lemma 1.

Theorem 1: Consider system (1) with input (2). If there exist a set of a family of non-negative functions $V_i : [nM + 1, (n + 1)M] \times \mathbb{R}^n \to \mathbb{R}_+$, $n = 0, 1, \ldots$, $i \in \mathcal{I}[1, N^M]$ satisfying (7), and exist scalars $0 < \alpha_i, \alpha_{i,j} < 1$ such that $\forall (i, j) \in \mathcal{I}[1, N^M] \times \mathcal{I}[1, N^M]$,

$$\Omega_i(k) \leq 0,\ \forall k = nM + 1, \ldots, (n + 1)M \tag{8}$$

$$\Theta_{i,j} \leq 0 \tag{9}$$

where $\Omega_i(k) = V_i(k + 1, x(k + 1)) - \alpha_i V_i(k, x(k)) - \frac{1-\alpha_i}{d^2}\omega^\top(k)\omega(k)$, $\Theta_{i,j} = V_j(nM + 1, x(nM + 1)) - \alpha_{i,j} V_i(nM, x(nM)) - \frac{1-\alpha_{i,j}}{d^2}\omega^\top(nM)\omega(nM)$, $n = 1, 2, \ldots$. Then system (1) is uniformly stable and we have $\exists i \in \mathcal{I}[1, N^M]$ such that $V_i(x(k)) \leq 1$ for all x_0 satisfying $V_i(0, x_0) \leq 1$, $\forall i \in \mathcal{I}[1, N^M]$.

Proof: First, we consider $\omega(k) = 0$ for stability. By (8), it ensures that

$$V_i(k + 1, x(k + 1)) - \alpha_i V_i(k, x(k)) \leq 0 \tag{10}$$

holds for $k = nM + 1, nM + 1, \ldots, (n + 1)M$.

Then, by (9), one has

$$V_j(nM + 1, x(nM + 1)) - \alpha_{i,j} V_i(nM, x(nM)) \leq 0 \tag{11}$$

Define a new function $\tilde{\sigma} : \mathbb{N} \to \mathcal{I}[1, N^M]$ indicating the active M-step sequence, and choose a Lyapunov function candidate as $V_{\tilde{\sigma}(k)}(k, x(k))$. According to Proposition 1, and together with (10) and (11) with the fact $0 < \alpha_i, \alpha_{i,j} < 1$, it leads to

$$V_{\tilde{\sigma}(k+1)}(k + 1, x(k + 1)) - V_{\tilde{\sigma}(k)}(k, x(k)) < 0 \tag{12}$$

Combined with (7), the stability can be established by standard Lyapunov theorem.

Furthermore, in presence of input $\omega(k)$, (8) yields that

$$\begin{aligned} V_i(k + 1, x(k + 1)) - \alpha_i V_i(k, x(k)) &\leq \frac{1 - \alpha_i}{d^2}\omega^\top(k)\omega(k) \\ &\leq 1 - \alpha_i \end{aligned} \tag{13}$$

which implies $V_i(k+1, x(k+1)) - 1 \leq \alpha_i(V_i(k, x(k)) - 1)$.

Similarly, (9) can lead to

$$V_j(nM + 1, x(nM + 1)) - \alpha_{i,j} V_i(nM, x(nM)) \leq 1 - \alpha_i \tag{14}$$

holds for $n = 1, 2, \ldots$, which implies

$$V_j(nM + 1, x(nM + 1)) - 1 \leq \alpha_{i,j}(V_i(nM, x(nM)) - 1) \tag{15}$$

For any $k \in \mathbb{N}$, we have

$$\begin{aligned} &V_{\tilde{\sigma}(k)}(k) - 1 \\ \leq &\alpha_{\tilde{\sigma}(k-1)}(V_{\tilde{\sigma}(k-1)}(k - 1) - 1) \\ \leq &\alpha_{\tilde{\sigma}(k-1)} \cdots \alpha_{\tilde{\sigma}(nM+1)}(V_{\tilde{\sigma}(nM)}(nM + 1) - 1) \\ \leq &\alpha_{\tilde{\sigma}(k-1)} \cdots \alpha_{\tilde{\sigma}(nM+1)}\alpha_{\tilde{\sigma}(nM+1),\tilde{\sigma}(nM)}(V_{\tilde{\sigma}(nM)}(nM) - 1) \\ \leq &\alpha_{\tilde{\sigma}(k-1)} \cdots \alpha_{\tilde{\sigma}(nM)}\alpha_{\tilde{\sigma}(nM),\tilde{\sigma}(nM-1)} \cdots \alpha_{\tilde{\sigma}(0)}(V_{\tilde{\sigma}(0)}(0) - 1) \end{aligned}$$

Due to $0 < \alpha_i, \alpha_{i,j} < 1$ and $V_i(0, x_0) \leq 1$, $\forall i \in \mathcal{I}[1, N^M]$, it ensures $V_{\tilde{\sigma}(k)}(k) - 1 \leq 0$, $\forall k \in \mathbb{N}$. Because $\tilde{\sigma}(k)$ is an arbitrary signal selecting value in $\mathcal{I}[1, N^M]$, it implies $\exists i \in \mathcal{I}[1, N^M] \Rightarrow V_i(k, x) \leq 1$. ■

Remark 3: If we particularly let $M = 1$, Condition (8) is reduced to

$$V_i(n+1, x(n+1)) - \alpha_i V_i(n, x(n)) - \frac{1 - \alpha_i}{d^2}\omega^\top(n)\omega(n) \leq 0 \tag{16}$$

and (9) can be rewritten to

$$V_j(n+1, x(n+1)) - \alpha_{i,j} V_i(n, x(n)) - \frac{1 - \alpha_{i,j}}{d^2}\omega^\top(n)\omega(n) \leq 0 \tag{17}$$

It is noted that (16) can be absorbed in (17) by just letting $\alpha_{i,i} = \alpha_i$. It can be seen that (17) is exactly the condition (4) in Lemma 1. Therefore, Theorem 1 covers previous result stated by Lemma 1, namely Lemma 1 is a particular case which can be recovered by Theorem 1 with $M = 1$.

B. Reachable Set Estimation

In this subsection, the reachable set estimation for discrete-time switched linear system will be investigated. Based on Theorem 1, the following result can be obtained.

Theorem 2: Consider system (1) with input (2). If there exist matrices $P_{i,m} \succ 0$, $m \in \mathcal{I}[1, M]$, $i \in \mathcal{I}[1, N^M]$ and scalars $0 < \alpha_i < 1$, $0 < \alpha_{i,j} < 1$ such that $\forall (i, j) \in \mathcal{I}[1, N^M] \times \mathcal{I}[1, N^M]$,

$$\begin{bmatrix} A_{i_m}^\top P_{i,m+1} A_{i_m} - \alpha_i P_{i,m} & A_{i_m}^\top P_{i,m} B_{i_m} \\ * & B_{i_m}^\top P_{i,m} B_{i_m} - \frac{1-\alpha_i}{d^2} I \end{bmatrix} \preceq 0$$

$$m = 1, 2, \ldots, M - 1 \tag{18}$$

$$\begin{bmatrix} A_{i_M}^\top P_{j,1} A_{i_M} - \alpha_{i,j} P_{i,M} & A_{i_M}^\top P_{j,1} B_{i_M} \\ * & B_{i_M}^\top P_{j,1} B_{i_M} - \frac{1-\alpha_{i,j}}{d^2} I \end{bmatrix} \preceq 0 \tag{19}$$

then system (1) is uniformly stable and the reachable set $\mathcal{R}_x$ can be over approximated by

$$\tilde{\mathcal{R}}_x \triangleq \bigcup_{m\in\mathcal{I}[1,M],i\in\mathcal{I}[1,N^M]} \mathcal{E}(P_{i,m}) \tag{20}$$

Proof: Choosing an M-step clock-dependent switched Lyapunov function in the following quadratic form

$$V_{\tilde{\sigma}(k)}(k,x(k)) = x^\top(k)P_{\tilde{\sigma}(k),k-nM+1}x(k),\ n=0,1,\ldots \tag{21}$$

where $\tilde{\sigma}(k)$ indicating the active M-step sequence defined same as in (12).

Suppose $\tilde{\sigma}(k) = i$, $k \in [nM+1,(n+1)M]$ and denote $\Omega_i(k) = V_i(k+1,x(k+1)) - \alpha_i V_i(k,x(k)) - \frac{1-\alpha_i}{d^2}\omega^\top(k)\omega(k)$, and thus the M-step sequence $\mathcal{S}_i^M = \{i_1,i_2,\ldots,i_M\}$. Along the system evolution, we can obtain

$$\Omega_i(k) = \xi^\top(k)\Xi_{i,m}\xi(k)$$

where $m\in\mathcal{I}[1,M]$, $\xi(k) = [x^\top(k),\omega^\top(k)]^\top$ and $\Xi_{i,m} = \begin{bmatrix} A_{i_m}^\top P_{i,m+1}A_{i_m} - \alpha_i P_{i,m} & A_{i_m}^\top P_{i_m}B_{i_m} \\ * & B_{i_m}^\top P_{i,m}B_{i_m} - \frac{1-\alpha_i}{d^2}I \end{bmatrix}$.

Moreover, assume $\tilde{\sigma}(nM) = i$ and $\tilde{\sigma}(nM+1) = j$, and let $\Theta_{i,j} = V_j(nM+1,x(nM+1)) - \alpha_{i,j}V_i(nM,x(nM)) - \frac{1-\alpha_{i,j}}{d^2}\omega^\top(nM)\omega(nM)$, the following derivation can be obtained for the transition from instant nM to $nM+1$.

$$\Theta_{i,j} = \xi^\top(nM)\Pi_{i,j}\xi(nM)$$

where

$$\Pi_{i,j} = \begin{bmatrix} A_{i_M}^\top P_{j,1}A_{i_M} - \alpha_{i,j}P_{i_M} & A_{i_M}^\top P_{j,1}B_{i_M} \\ * & B_{i_M}^\top P_{j,1}B_{i_M} - \frac{1-\alpha_{i,j}}{d^2}I \end{bmatrix}$$

By (18) and (19), it can be ensured that $\Omega_i(k) \le 0$, $\forall k = nM+1,\ldots,(n+1)M$, $\forall n = 1,2,\ldots$ and $\Theta_{i,j} \le 0$.

According to Theorem 1, for the case of $x_0 = 0$, we have $\exists i \in \mathcal{I}[1,N^M]$ such that $V_i(x(k)) \le 1$. Therefore, the state $x(k)$ satisfies $x \in \{x \mid x^\top P_{i,m}x \le 1, m \in \mathcal{I}[1,M], i \in \mathcal{I}[1,N^M]\} = \bigcup_{m\in\mathcal{I}[1,M],i\in\mathcal{I}[1,N^M]}\mathcal{E}(P_{i,m})$, which is exactly the set (20). The proof is complete. ■

Remark 4: Theorem 2 can be viewed as an improved version for Lemma 2, if we enforce $M = 1$ in Theorem 2, $P_{i,m}$, $m\in\mathcal{I}[1,M]$, $i\in\mathcal{I}[1,N^M]$, becomes P_i, $i\in\mathcal{I}[1,N]$. Inequalities (18) and (19) can be rewritten to

$$\begin{bmatrix} A_i^\top P_jA_i - \alpha_{i,j}P_i & A_i^\top P_jB_i \\ * & B_i^\top P_jB_i - \frac{1-\alpha_{i,j}}{d^2}I \end{bmatrix} \preceq 0$$

which is (5) in Lemma 2.

Remark 5: The set $\tilde{\mathcal{R}}_x$ is usually expected to be as small as possible to achieve a precise estimate of reachable set $\mathcal{R}_y$. In [24], several methods have been proposed to minimize the bounding ellipsoids, which can be also employed in our paper. In order to make a clear comparison with [24], we consider the method associated to the following constraint

$$P_{i,m} \succeq \epsilon I,\ \epsilon > 0,\ \forall m\in\mathcal{I}[1,M],\ \forall i\in\mathcal{I}[1,N^M] \tag{22}$$

which implies that $\epsilon x^\top(k)x(k) \le x^\top(t)P_{i,m}x(k) \le 1$, namely $x(t) \in \tilde{\mathcal{R}}_x \triangleq \bigcup_{m\in\mathcal{I}[1,M],i\in\mathcal{I}[1,N^M]}\mathcal{E}(P_{i,m}) \subseteq \mathcal{B}(0,1/\sqrt{\epsilon})$, $\forall k\in\mathbb{R}_{\ge 0}$, so we have to maximize ϵ to obtain a smallest ball $\mathcal{B}(0,1/\sqrt{\epsilon})$ as

$$\begin{aligned} &\max\ \epsilon \\ &\text{s.t. } (18),\ (19) \text{ and } (22) \end{aligned} \tag{23}$$

TABLE I
COMPUTATIONAL COMPLEXITY OF THEOREM 2

	Number of variables	Size of LMIs
Theorem 2	$\frac{n(n+1)MN^M}{2}$	$n(N^{2M}+MN^M)$

Moreover, due to the existence of tuning parameters α_i and $\alpha_{i,j}$, the result in Theorem 2 and corresponding optimization problem (23) are not standard LMI problems, they are bilinear matrix inequality (BMI) problems and known to be NP-hard. Fortunately, several algorithms are available to solve BMI problems such as the iterative linear matrix inequality (ILMI) approach in [26], [27], or using numerical optimization algorithms, such as program `fminsearch` [20] or genetic algorithm (GA) [24] in the optimization toolbox of Matlab.

Remark 6: Although $M > 1$ will reduce the conservativeness, the price to pay is the increase of computational complexity. The number of LMIs and involved decision variables grows as M is increased. The computation complexities are listed in Table I.

C. Some Discussions for Stability Analysis

It should be noted that the stability analysis result of switched system (1) with input $\omega(k) = 0$ is actually included in the previous reachable set estimation solution. As what has been shown in previous section, our reachable set estimation yields less conservativeness than that in [24] which is essentially based on switched Lyapunov function approach in [25] . In fact, by introducing the concept of M-step sequence, a less conservative stability analysis result can be obtained as well in contrast to the well known stability criterion proposed in [25] on basis of switched Lyapunov function approach.

The following corollary can be viewed as an improvement for the classical switched Lyapunov function approach in stability analysis.

Corollary 1: Consider switched system (1) with $\omega(k) = 0$, if there exist MN^M symmetric matrices $P_{i,m} \succ 0$, $m\in\mathcal{I}[0,M]$, $i\in\mathcal{I}[1,N^M]$ such that the following inequalities hold for $\forall i,j\in\mathcal{I}[1,N^M]$, $\forall m\in\mathcal{I}[1,M]$,

$$A_{i_{m+1}}^\top P_{i,m+1}A_{i_m} - P_{i,m} \prec 0,\ m = 1,2,\ldots,M-1 \tag{24}$$

$$A_{i_M}^\top P_{j,1}A_{i_M} - P_{i,M} \prec 0 \tag{25}$$

then switched system (1) is globally uniformly asymptotically stable.

Proof: The proof can be obtained by the guidelines in Theorems 1 and 2, which is omitted here. ■

Remark 7: Corollary 1 can be viewed as an improved result over switched Lyapunov function approach for switched system (1). By letting $M = 1$, conditions (24) and (25) can be rewritten to

$$A_i^\top P_j A_i - P_i \prec 0,\ i, j \in \mathcal{I}[1, N] \quad (26)$$

where $P_i \succ 0$, $i \in \mathcal{I}[1, N]$. This result is exactly the Theorem 2 in [25], which means that the switched Lyapunov function approach is a special case of Corollary 1 as $M = 1$. Corollary 1 with $M \geq 2$ is able to yield less conservativeness in stability analysis, which can be shown by a numerical example later.

IV. Example

Example 1: Consider a switched system with two modes with the following system matrices

$$A_1 = \begin{bmatrix} 0 & 0.7 \\ -0.2 & -0.6 \end{bmatrix},\ B_1 = \begin{bmatrix} 0.2 \\ -0.4 \end{bmatrix}$$
$$A_2 = \begin{bmatrix} -0.6 & 0.4 \\ -0.7 & 0.2 \end{bmatrix},\ B_2 = \begin{bmatrix} -0.6 \\ 0.4 \end{bmatrix}$$

The disturbance $\omega(k)$ satisfies $\omega(k) \in \mathcal{W} \triangleq \{\omega \in \mathbb{R}^{n_\omega} \mid \omega^\top \omega \leq 1\}$. In order to compare our approach with that in [24], we first use Lemma 2 to obtain the reachable set estimation by maximizing ϵ in optimization (23). The GA is used to search for optimized α_i, $i \in \mathcal{I}[1, 2]$. The population is set to be 50. After 100 generations, the optimal $\epsilon = 0.04057$, which is shown in Fig. 1.

On the other hand, with same population and generation, Theorem 2 with $M = 2$ reaches a larger ϵ as $\epsilon = 0.05618$, which obviously is a less conservative result. The update of ϵ at each generation is illustrated in Fig. 1, which has a slower convergent rate but a better optimized result. The slower convergent rate is basically because more variables $\alpha_{i,j}$, $i, j \in \mathcal{I}[1, 4]$, are introduced in the optimization problem. The union of bounding ellipsoids are depicted in Fig. 2 by solid blue lines. For the purpose of showing the advantage of our approach, we present Fig. 3 to clearly compare Theorem 2 and Lemma 2, in which the estimation by Theorem 2 is more precise than by Lemma 2. In Figs. 2 and 3, the state trajectories are generated with arbitrary switching signal and disturbance $\omega(k)$ uniformly distributed over $[-1, 1]$.

Example 2: In this example, we will show the less conservativeness of M-step method in the stability point of view. Let us consider the system (1) with matrices $A_i = e^{B_i T}$, where

$$B_1 = \begin{bmatrix} 0 & 1 \\ -10 & -1 \end{bmatrix},\ B_2 = \begin{bmatrix} 0 & 1 \\ -0.1 & -4 \end{bmatrix} \quad (27)$$

Letting $T = 0.1$, and using switched Lyapunov function approach in [25] (also viewed as $M = 1$ in our M-step sequence approach), it can be found that the LMI problem is not feasible, so that the globally uniformly asymptotically stability cannot be determined by the approach in [25]. Moreover, by applying the method in [28], the minimum admissible dwell time is computed as 2, which also indicates that the globally uniformly asymptotically stability of switched system (1) cannot be ascertained for the case

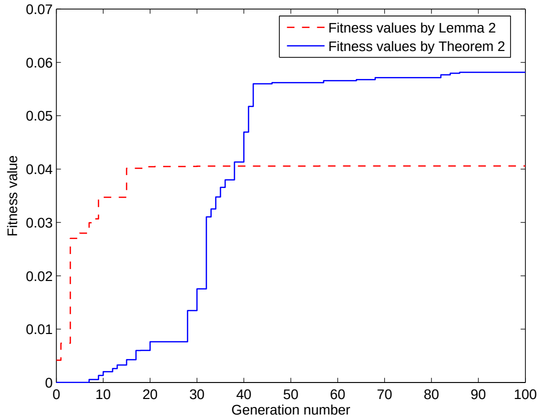


Fig. 1. Fitness function value along with generations.

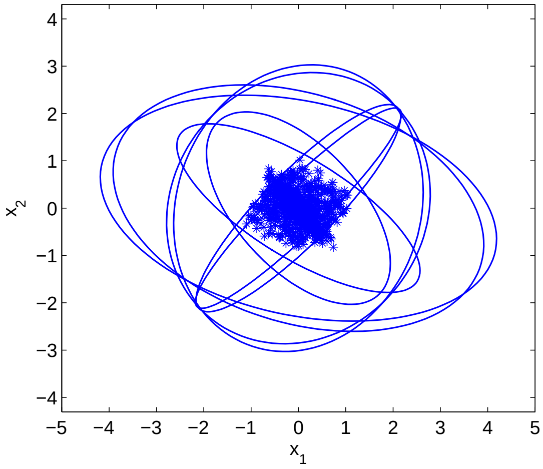


Fig. 2. Bounding ellipsoids by Theorem 2.

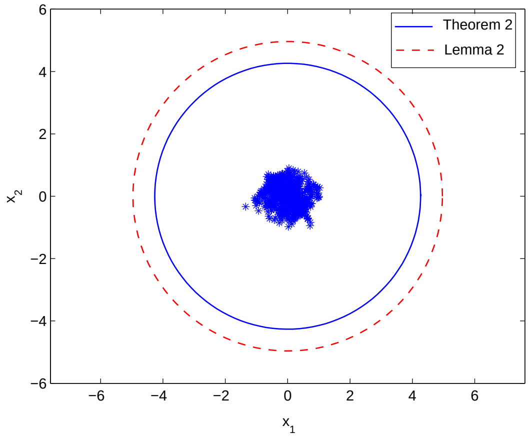


Fig. 3. Bounding circles by Lemma 2 and Theorem 2.

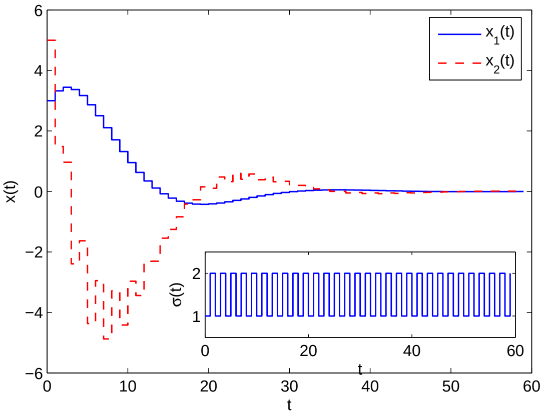


Fig. 4. State response under switching occurring at each time instant.

of arbitrary switching, for which the minimum dwell time should be 1.

However, if we increase M by just letting $M = 2$ in the M-step sequence method proposed in this paper, the feasibility of the corresponding LMI problems can be established, which is sufficient to guarantee that the system is globally uniformly asymptotically stable under arbitrary switching. The convergent state evolution is shown by the following simulation result in Fig. 4, where the extreme switching behavior, i.e., the switching occurs at each time instant, is adopted, and the initial state is assumed to be $x_0 = [3 \;\; 5]^\top$.

V. Conclusions

The reachable set estimation problem for discrete-time switched system has been investigated in this paper. A novel conception called M-step sequence is introduced to solve the reachable set estimation problem, it is shown that the proposed approach covers the previous result which is based on switched Lyapunov function, and thus has less conservativeness. In addition, some discussions are given for stability analysis for discrete-time switched system in the framework of M-step sequence. Finally, numerical examples are given to show the theoretical findings in this paper.

References

[1] R. DeCarlo, M. S. Branicky, S. Pettersson, and B. Lennartson, "Perspectives and results on the stability and stabilizability of hybrid systems," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 1069–1082, 2000.

[2] D. Liberzon, *Switching in Systems and Control*. Springer Science & Business Media, 2012.

[3] R. Shorten, F. Wirth, O. Mason, K. Wulff, and C. King, "Stability criteria for switched and hybrid systems," *SIAM Review*, vol. 49, no. 4, pp. 545–592, 2007.

[4] H. Lin and P. J. Antsaklis, "Stability and stabilizability of switched linear systems: a survey of recent results," *IEEE Transactions on Automatic control*, vol. 54, no. 2, pp. 308–322, 2009.

[5] M. Johansson, A. Rantzer, *et al.*, "Computation of piecewise quadratic Lyapunov functions for hybrid systems," *IEEE transactions on automatic control*, vol. 43, no. 4, pp. 555–559, 1998.

[6] M. Margaliot and G. Langholz, "Necessary and sufficient conditions for absolute stability: the case of second-order systems," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 50, no. 2, pp. 227–234, 2003.

[7] S. Pettersson and B. Lennartson, "Stabilization of hybrid systems using a min-projection strategy," in *American Control Conference, 2001. Proceedings of the 2001*, vol. 1, pp. 223–228, IEEE, 2001.

[8] A. S. Morse, "Supervisory control of families of linear set-point controllers part i. exact matching," *IEEE Transactions on Automatic Control*, vol. 41, no. 10, pp. 1413–1431, 1996.

[9] J. P. Hespanha and A. S. Morse, "Stability of switched systems with average dwell-time," in *Decision and Control, 1999. Proceedings of the 38th IEEE Conference on*, vol. 3, pp. 2655–2660, IEEE, 1999.

[10] W. Xiang and J. Xiao, "Stabilization of switched continuous-time systems with all modes unstable via dwell time switching," *Automatica*, vol. 50, no. 3, pp. 940–945, 2014.

[11] W. Xiang, "On equivalence of two stability criteria for continuous-time switched systems with dwell time constraint," *Automatica*, vol. 54, pp. 36–40, 2015.

[12] W. Xiang, "Necessary and sufficient condition for stability of switched uncertain linear systems under dwell-time constraint," *IEEE Transactions on Automatic Control*, vol. 61, no. 11, pp. 3619–3624, 2016.

[13] L. Allerhand and U. Shaked, "Robust stability and stabilization of linear switched systems with dwell time," *IEEE Transactions on Automatic Control*, vol. 56, no. 2, pp. 381–386, 2011.

[14] C. Briat, "Convex lifted conditions for robust ℓ_2-stability analysis and ℓ_2-stabilization of linear discrete-time switched systems with minimum dwell-time constraint," *Automatica*, vol. 50, no. 3, pp. 976–983, 2014.

[15] L. Zhang and H. Gao, "Asynchronously switched control of switched linear systems with average dwell time," *Automatica*, vol. 46, no. 5, pp. 953–958, 2010.

[16] L. Zhang and P. Shi, "Stability, ℓ_2-gain and asynchronous control of discrete-time switched systems with average dwell time," *IEEE Transactions on Automatic Control*, vol. 54, no. 9, pp. 2192–2199, 2009.

[17] E. Fridman, A. Pila, and U. Shaked, "Regional stabilization and $\mathcal{H}_\infty$ control of time-delay systems with saturating actuators," *International Journal of Robust and Nonlinear Control*, vol. 13, no. 9, pp. 885–907, 2003.

[18] J. Lygeros, C. Tomlin, and S. Sastry, "Controllers for reachability specifications for hybrid systems," *Automatica*, vol. 35, no. 3, pp. 349–370, 1999.

[19] C. Durieu, E. Walter, and B. Polyak, "Multi-input multi-output ellipsoidal state bounding," *Journal of optimization theory and applications*, vol. 111, no. 2, pp. 273–303, 2001.

[20] E. Fridman and U. Shaked, "On reachable sets for linear systems with delay and bounded peak inputs," *Automatica*, vol. 39, no. 11, pp. 2005–2010, 2003.

[21] Z. Feng and J. Lam, "An improved result on reachable set estimation and synthesis of time-delay systems," *Applied Mathematics and Computation*, vol. 249, pp. 89–97, 2014.

[22] B. Zhang, J. Lam, and S. Xu, "Reachable set estimation and controller design for distributed delay systems with bounded disturbances," *Journal of the Franklin Institute*, vol. 351, no. 6, pp. 3068–3088, 2014.

[23] Z. Feng and J. Lam, "On reachable set estimation of singular systems," *Automatica*, vol. 52, pp. 146–153, 2015.

[24] Y. Chen, J. Lam, and B. Zhang, "Estimation and synthesis of reachable set for switched linear systems," *Automatica*, vol. 63, pp. 122–132, 2016.

[25] J. Daafouz, P. Riedinger, and C. Iung, "Stability analysis and control synthesis for switched systems: a switched Lyapunov function approach," *IEEE transactions on automatic control*, vol. 47, no. 11, pp. 1883–1887, 2002.

[26] Y.-Y. Cao, J. Lam, and Y.-X. Sun, "Static output feedback stabilization: an ILMI approach," *Automatica*, vol. 34, no. 12, pp. 1641–1645, 1998.

[27] Z. Shu and J. Lam, "An augmented system approach to static output-feedback stabilization with $\mathcal{H}_\infty$ performance for continuous-time plants," *International Journal of Robust and Nonlinear Control*, vol. 19, no. 7, pp. 768–785, 2009.

[28] W. Xiang and J. Xiao, "Convex sufficient conditions on asymptotic stability and ℓ_2 gain performance for uncertain discrete-time switched linear systems," *IET Control Theory & Applications*, vol. 8, no. 3, pp. 211–218, 2014.

Event-Triggered Control for Continuous-Time Switched Linear Systems

Weiming Xiang [†] Taylor T. Johnson [†]

January 12, 2017

Abstract

The event-triggered control problem for switched linear system is addressed in this paper. The periodical sampling scheme and event-triggering condition are incorporated in the closed-loop. The feedback control updates its value only at sampling instants as long as event-triggering condition is satisfied as well. In addition, the switchings are only allowed to occur at sampling instants and meanwhile the switching condition is satisfied. Three equivalent sufficient conditions are proposed to ensure the asymptotic stability of switched systems. In particular, one condition has a promising feature of affineness in system matrices, and as a consequence, it is extended to robust sampling case and $\mathcal{L}_2$-gain analysis. Several examples are provided to illustrate our results.

Keywords: Asymptotic stability, event-triggered control, $\mathcal{L}_2$ gain, switched systems

1 Introduction

Switched systems have emerged as an important subclass of hybrid systems and represent a very active area of current research in the field of control systems [1–3]. A switched system is composed of a family of continuous or discrete-time subsystems, described by differential or difference equations, respectively, along with a switching rule governing the switching amongst the subsystems. The motivation for studying switched systems comes from the fact that switched system can be effectively used to model many practical systems that are inherently multi-model in the sense that several dynamic subsystem models are required to describe their behavior. For instance, the sampled data systems [4], networked control systems [5] and event-triggered systems [6] can be modeled as switched systems. Generally, the stability and stabilization problems are the main concerns in the field of switched systems. It has been proved that Lyapunov function techniques are effective to deal with stability and stabilization problems for switched systems, for example [7–9]. Combining multiple Lyapunov function (MLF), the dwell time and average dwell time properties of relatively slowly switched systems have been investigated in the corresponding switched systems [10–12]. For more details on the recent advances in the area, the readers are referred to the surveys [2], and the references cited therein.

On the other hand, the periodic and aperiodic control strategies are presented as the most prevailing control approaches on digital platforms. Typically, the control executes periodically in the closed-loop and the system can be analyzed by the well-developed sampled-data system theory. As a further

[†] Authors are with the Department of Electrical Engineering and Computer Science, Vanderbilt University, Nashville, TN 37212 USA. Email: Weiming Xiang (xiangwming@gmail.com), Taylor T. Johnson (taylor.johnson@gmail.com).

improvement of traditional sampled-data system, the event-triggered control system is introduced, see for example the theory work [13–16], and numerous applications [17–20]. In the framework of event-triggered control, the control executions are generated by well-designed event-triggering condition. In comparison with sampled-data scheme, the event-triggered control which is a typical aperiodic one is capable of significantly reducing the number of control task executions, while retaining a satisfactory closed-loop performance. Though the event-triggered control can offers some clear advantages with respect to periodic control such as in handling energy, computation, and communication constraints but it also introduces some new theoretical and practical problems. The detailed advantages and challenges introduced by the event-triggered control can be found in the survey paper [21].

In this paper, we consider a class of periodic event-triggered control for switched linear systems. The periodic event-triggering condition allows the coexistence of periodic sampling scheme and event-triggering condition for the control executions. Moreover, this blending strategy also determines the occurrence of switching behaviors, in other words, the switching only occurs at sampling instants as long as the switching condition is satisfied. Three stability criteria are proposed for event-triggered switched system in this paper, and they are proved to be basically equivalent. The first one is derived by analyzing the evolution of state at sampling instant, however, it is not convenient to extend to further problems such as robust sampling and $\mathcal{L}_2$-gain analysis. Then, a sampling-dependent approach is proposed, which actually is not numerically tractable since it has infinitely many values to check. Thus, a discretized method to equivalently convert the sampling-dependent condition into a numerically tractable condition. Based on this numerically tractable condition, the extensions to robust sampling case and $\mathcal{L}_2$-gain analysis are made.

The remainder of this paper is organized as follows: The event-triggered switched system model is given in Section 2. The main result, three equivalent stability criteria are presented in Section 3. Extensions to robust sampling case and $\mathcal{L}_2$-gain analysis are studied in Section 4 and Section 5, respectively. Conclusions are given Section 6.

Notations: $\mathbb{N}$ represents the set of natural numbers, $\mathbb{R}$ denotes the field of real numbers, $\mathbb{R}^+$ is the set of nonnegative real numbers, and $\mathbb{R}^n$ stands for the vector space of all n-tuples of real numbers, $\mathbb{R}^{n\times n}$ is the space of $n \times n$ matrices with real entries. The set $\mathbb{M}_c^n$ consists of all matrices $\Phi \in \mathbb{R}^{n\times n}$ with nonnegative off diagonal elements $\phi_{ji} \geq 0$, $i \neq j$, satisfying $\sum_{j=1}^N \phi_{ji} = 0$, which implies that $\phi_{ii} \leq 0$. The set $\mathbb{M}_d^n$ consists of all matrices $\Pi \in \mathbb{R}^{n\times n}$ with nonnegative elements $\pi_{ji} \geq 0$ satisfying the normalization constraints $\sum_{j=1}^N \pi_{ji} = 1$. $\|\cdot\|$ stands for Euclidean norm. The notation $A \succ 0$ means A is real symmetric and positive definite. $A \succ B$ means that $A - B \succ 0$. $A^\top$ denotes the transpose of A. In addition, in symmetric block matrices, we use $*$ as an ellipsis for the terms that are induced by symmetry and diag$\{\cdots\}$ stands for a block-diagonal matrix. I denotes the unit matrix and 0 stands for the zero elements in matrix with appropriate dimensions. We define $x(t_k^+) = \lim_{t\to t_k^+} x(t)$ and $x(t_k^-) = \lim_{t\to t_k^-} x(t)$. For a matrix function $F : [a,b] \to \mathbb{R}^{n\times n}$, its upper right Dini derivative is defined by $\mathcal{D}^+F(x) \triangleq \lim_{h\to 0^+} \sup \frac{F(x+h)-F(x)}{h}$. In the rest of this work, we will make extensive uses of the following matrix expressions:

$$\begin{aligned}
\mathscr{C}(A,P) &= A^\top P^\top + PA \\
\mathscr{D}(A,P(t)) &= \mathscr{C}(A,P(t)) + \mathcal{D}^+P(t) \\
\mathscr{D}_1(A,P,Q,\delta) &= \mathscr{C}(A,P) + (P-Q)/\delta \\
\mathscr{D}_2(A,P,Q,\delta) &= \mathscr{C}(A,Q) + (P-Q)/\delta \\
\mathscr{E}(A,J,P,Q,t) &= e^{A^\top t} J^\top P J e^{At} - Q
\end{aligned}$$

2 Event-Triggered Switched Control System

Consider the continuous-time switched linear system in the following form:

$$\dot{x}(t) = A_{\sigma(t)}x(t) + B_{\sigma(t)}u(t) + E_{\sigma(t)}\omega(t) \tag{1}$$

$$y(t) = C_{\sigma(t)}x(t) + D_{\sigma(t)}\omega(t) \tag{2}$$

where $x(t), x_0 \in \mathbb{R}^n$ are the state of the system and the initial condition, respectively. $u(t) \in \mathbb{R}^{n_u}$ is the input and $\omega(t) \in \mathbb{R}^{n_\omega}$ is the exogenous disturbance. $y(t) \in \mathbb{R}^{n_y}$ is the controlled output. The switching function $\sigma : \mathbb{R}^+ \to \mathcal{N} \triangleq \{1, 2, \ldots, N\}$ defines the switching actions, where N is the number of subsystems.

In this paper, we consider a periodic event-triggered control strategy for switched system (1)–(2) for the sake of taking advantages of both periodic sampled-data and event-triggered control, which means the system state $x(t)$ is only measured at the periodic sampling times for generating the control input, computing the switching function output and verifying the event-triggering condition. In a periodic sampling implementation, the values of the system state are available for a time sequence $\mathcal{S} \triangleq \{t_k\}_{k\in\mathbb{N}}$, where t_0 is the initial time and t_k, $k \in \mathbb{N} \setminus \{0\}$, are the sampling times, which are periodic in the sense that $t_k = kT_s$, $k \in \mathbb{N}$, for some properly chosen sampling interval $T_s > 0$. With this sampling setting, the sampled switching signal is

$$\sigma(t) = \hat{\sigma}(t),\ t \in (t_k, t_{k+1}] \tag{3}$$

where $\hat{\sigma}(t)$, $t \in (t_k, t_{k+1}]$, is determined by

$$\hat{\sigma}(t) = \begin{cases} \sigma(t_k) & \sigma(t_k) \neq \hat{\sigma}(t_k) \\ \hat{\sigma}(t_k) & \sigma(t_k) = \hat{\sigma}(t_k) \end{cases} \tag{4}$$

The sampled switching signal (3)–(4) implies the switching decisions are only made at sampling instant t_k. The value of $\sigma(t)$ only changes at sampling instant t_k if $\sigma(t_k) \neq \hat{\sigma}(t_k)$, otherwise it holds its most recent value. It worth mentioning that since the switching function (3) only activates at each sampling time t_k, $k \in \mathbb{N}$, it can be interpreted that a dwell time constraint $t_{k+1} - t_k \geq T_s$, $\forall k \in \mathbb{N}$ is imposed on the switching signal. This dwell time constraint obviously prevents the switching actions from chattering phenomenon or Zeno phenomenon, since the switching frequency is restricted to have an upper bound equals to $1/T_s$. In [22], a modified min-switching law with dwell time constraint is proposed to avoid the chattering behavior owe to the dwell time constraint. However, it requires accessing the system state and monitoring the state-dependent switching rule continuously, which is not allowed in the sampled-data setting proposed in this paper, since the system state $x(t)$ is obtained only at sampling instants.

In addition, we also take the sampled-data feedback controller into account. In a conventional periodic sampled-data control scheme, the following mode-dependent state feedback controller is often considered

$$u(t) = K_{\sigma(t)}\hat{x}(t),\ t \in \mathbb{R}^+ \tag{5}$$

where K_i, $i \in \mathcal{N}$ are the already designed feedback gains for subsystems, and $\hat{x}(t)$, $t \in (t_k, t_{k+1}]$, is defined by

$$\hat{x}(t) = x(t_k),\ t \in (t_k, t_{k+1}] \tag{6}$$

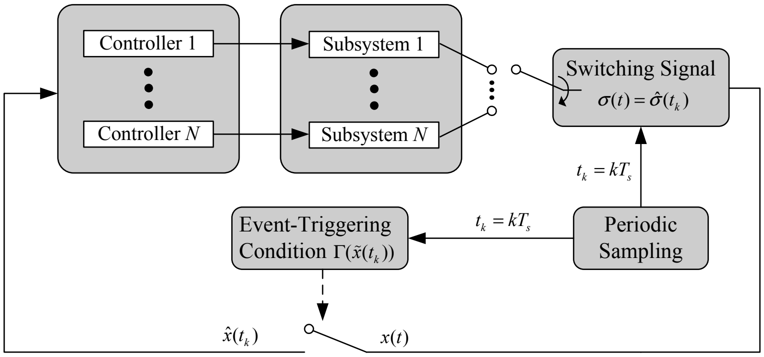


Figure 1: General scheme of periodic event-triggered switched control system

In order to obtain a complete model of system (1)–(2) with the periodic sampling setting (3) and (5), we let $\tilde{x}(t) = [x(t) \;\; \hat{x}(t)]^\top$ and obtain the following system

$$\dot{\tilde{x}}(t) = \tilde{A}_{\sigma(t)}\tilde{x}(t) + E_{\sigma(t)}\omega(t),\ t \in \mathbb{R}^+ \setminus \mathcal{S} \tag{7}$$

$$\tilde{x}(t_k^+) = J\tilde{x}(t_k^-),\ t_k \in \mathcal{S} \tag{8}$$

$$y(t) = \tilde{C}_{\sigma(t)}\tilde{x}(t) + \tilde{D}_{\sigma(t)}\omega(t) \tag{9}$$

where $\sigma(t)$ evolves according to (3) and

$$\tilde{A}_i = \begin{bmatrix} A_i & B_iK_i \\ 0 & 0 \end{bmatrix},\ \tilde{E}_i = \begin{bmatrix} E_i \\ 0 \end{bmatrix},\ J = \begin{bmatrix} I & 0 \\ I & 0 \end{bmatrix},\ \tilde{C}_i = \begin{bmatrix} C_i & 0 \end{bmatrix},\ \tilde{D}_i = D_i$$

Further considering the event-triggered controller, the state measurements are transmitted over a communication network and the control values are updated only when certain event-triggering conditions are satisfied, the controller is given in the following form

$$u(t) = K_{\sigma(t)}\hat{x}(t),\ t \in \mathbb{R}^+ \tag{10}$$

where $\hat{x}(t)$ is a left-continuous signal, given for $t \in (t_k, t_{k+1}]$, $k \in \mathbb{N}$, and modifies the (6) as

$$\hat{x}(t) = \begin{cases} x(t_k), & \Gamma(x(t_k), \hat{x}(t_k)) > 0 \\ \hat{x}(t_k), & \Gamma(x(t_k), \hat{x}(t_k)) \leq 0 \end{cases} \tag{11}$$

with an event-triggering function $\Gamma : \mathbb{R}^{2n} \to \mathbb{R}$. The value $\hat{x}(t_k)$ stands for the valid value for the controller at sampling time t_k and through the successive interval $[t_k, t_{k+1})$, which is determined by the event-triggering function Γ. If $\Gamma(x(t_k), \hat{x}(t_k)) \leq 0$, the state $\hat{x}(t_k)$ holds as its most recent value, and in the case of $\Gamma(x(t_k), \hat{x}(t_k)) > 0$, the state $x(t_k)$ is transmitted over the network to the controller and $\hat{x}(t_k)$ is updated accordingly. The general scheme of event-triggered switched control system with periodic sampling setting is illustrated in Figure 1.

In this paper, we focus on a class of quadratic event-triggering condition, that is, $\Gamma(x(t_k), \hat{x}(t_k))$ is in the following quadratic form

$$\Gamma(\tilde{x}(t_k)) = \tilde{x}^\top(t_k)Q\tilde{x}(t_k) \tag{12}$$

where $\tilde{x}(t_k) = [x^\top(t_k)\ \hat{x}^\top(t_k)]^\top$ and $Q \in \mathbb{R}^{2n\times 2n}$ is a symmetric matrix. Several event-triggering conditions can be written into the quadratic structure (12), for example the state-error based triggering condition $\Gamma(x(t_k), \hat{x}(t_k)) = \|\hat{x}(t_k) - x(t_k)\| - \Delta\, \|x(t_k)\|$, where $\Delta > 0$, can be expressed by (12) with

$$Q = \begin{bmatrix} (1-\Delta^2)I & -I \\ -I & I \end{bmatrix}$$

Other well-known event triggering conditions such as input-error based, Lyapunov function based conditions can be formalized by (12) as well, readers can refer to [6].

In summary, by modifying the periodic sampled-data system model (7)–(9), the event-triggered system model arrives at

$$\dot{\tilde{x}}(t) = \tilde{A}_{\sigma(t)}\tilde{x}(t) + \tilde{E}_{\sigma(t)}\omega(t),\ t \in \mathbb{R}^+ \setminus \mathcal{S} \tag{13}$$

$$\tilde{x}(t_k^+) = \begin{cases} J_1\tilde{x}(t_k^-), & \tilde{x}^\top(t_k^-)Q\tilde{x}(t_k^-) > 0 \\ J_2\tilde{x}(t_k^-), & \tilde{x}^\top(t_k^-)Q\tilde{x}(t_k^-) \le 0 \end{cases},\ t_k \in \mathcal{S} \tag{14}$$

$$y(t) = \tilde{C}_{\sigma(t)}\tilde{x}(t) + \tilde{D}_{\sigma(t)}\omega(t) \tag{15}$$

where J_1 is same as J in (8) and $J_2 = \mathrm{diag}\{I, I\}$.

By (13)–(15), one can see that the event-triggered switched control system can be expressed as a switched system with impulsive behaviors at switching instants. For the *passive* switching, that is the switching information is not available and the switching is supposed to possibly occur at every switching sampling instant, system (13)–(15) can be viewed to be under switching with a dwell time T_s. The results in [11, 23–25] for switched system with dwell time can be employed. However, if some *active* switching is considered, which means the switching rule is explicitly available to designed, the passive switching result could yields conservativeness, thus we should improve these results with the aid of the information of switching law. For the *active* switching considered in the remainder of paper, we adopt the well-known min-switching rule, which is described as below:

$$\sigma(t) = \arg\min_{i\in\mathcal{N}} \tilde{x}^\top(t)P_i\tilde{x}(t) \tag{16}$$

where $P_i \succ 0$, $i \in \mathcal{N}$, are matrices to be determined, see the results in [26, 27]. The corresponding sampled min-switching rule (16) is described as

$$\sigma(t) = \begin{cases} \arg\min_{i\in\mathcal{N}} \tilde{x}^\top(t_k^+)P_i\tilde{x}(t_k^+), & t_k \in \mathcal{S} \\ \sigma(t_k), & t \in (t_k, t_{k+1}) \end{cases} \tag{17}$$

The main aim of this paper is to provide analysis and design techniques for controller, sampling scheme, and event-triggering condition such that the system is stable with switching rule (17). In the following, the definition of globally asymptotic stability is presented.

Definition 1 *A function $\gamma : \mathbb{R}^+ \to \mathbb{R}^+$ is a $\mathcal{K}$ function if it is strictly increasing and $\gamma(0) = 0$, and also a function $\beta : \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R}^+$ is a $\mathcal{KL}$ function if for each fixed s the function $\beta(r, s)$ is a $\mathcal{K}$ function with respect to r, and for each fixed r the function $\beta(r, s)$ is decreasing with respect to s and $\beta(r, s) \to 0$ as $s \to 0$.*

The definition of globally uniformly asymptotic stability (GUAS) for system (13)–(15) is given below.

Definition 2 *The equilibrium $x = 0$ of system (13)–(15) with $\omega(t) = 0$ is GUAS under the switching signal $\sigma(t)$ if, for initial condition $\tilde{x}(t_0)$, there exists a class $\mathcal{KL}$ function β such that the solution of the system satisfies $\|\tilde{x}(t)\| \leq \beta(\|\tilde{x}(t_0)\|, t)$, $\forall t \in \mathbb{R}^+$.*

In the presence of input $\omega(t)$, the $\mathcal{L}_2$-gain performance of system (13)–(15) is formulated in the following.

Definition 3 *For $\gamma > 0$, system (13)–(15) is said to be GUAS with an $\mathcal{L}_2$-gain performance, if the following is satisfied:*

(1) System (13)–(15) is GUAS when $\omega(t) = 0$;

(2) Under zero initial conditions, the following inequality holds for all nonzero $\omega \in \mathcal{L}_2[0, \infty)$,

$$\int_{t_0}^{\infty} \|y(t)\|^2 \,\mathrm{d}t \leq \gamma^2 \int_{t_0}^{\infty} \|\omega(t)\|^2 \,\mathrm{d}t \tag{18}$$

where γ is called the $\mathcal{L}_2$-gain.

Before ending this section, a useful lemma is introduced.

Lemma 1 *For a matrix $A \in \mathbb{R}^{n\times n}$ and a scalar $T_s > 0$, there always exist a sufficiently large $M^* \in \mathbb{N} \setminus \{0\}$, a sufficiently small $\epsilon \in \mathbb{R}^+$ and matrices $P_m \in \mathbb{R}^{n\times n}$, $m = \{0, \ldots, M\}$, such that*

$$P_m \succ 0, \; m \in \{0, \ldots, M\} \tag{19}$$

$$\mathscr{D}_1(A, P_{m+1}, P_m, \delta) \prec 0, \; m \in \{0, \ldots, M-1\} \tag{20}$$

$$\mathscr{D}_2(A, P_{m+1}, P_m, \delta) \prec 0, \; m \in \{0, \ldots, M-1\} \tag{21}$$

where $\delta = T_s/M$, hold for any $M > M^*$, and P_m, $m = \{0, \ldots, M\}$, have the following form:

$$P_m = e^{-A^\top \delta_m} P_0 e^{-A\delta_m} - \int_0^{\delta_m} e^{-A^\top(\delta_m - t)} Y(t) e^{-A(\delta_m - t)(t)} \mathrm{d}t, \; m \in \{0, \ldots, M\} \tag{22}$$

where $\delta_m = mT_s/M$, $m = \{0, \ldots, M\}$, and $0 \prec Y(t) \prec \epsilon I$, $t \in [0, T_s]$.

Proof. See Appendix. □

In this section, the closed-loop of event-triggered switched linear system is modeled as a switched system with state update at switching instant, along with mixed time-dependent and state-dependent switching rules. In the next section, the stability analysis will be studied as the main result in this paper.

3 Stability Analysis for Event-Triggered Switched System

Motivated by the techniques used in [23, 24, 28–30] for switched systems, and [31] for time-delayed systems, the main result for the stability of event-triggered switched control system (13)–(15) is presented by the following theorem.

Theorem 1 *Consider event-triggered switched control system (13)–(15) with $\omega(t) = 0$, the following three statements are equivalent:*

(a) *There exist scalars $\mu_h > 0$, $h \in \{1,2\}$, a matrix $\Pi \in \mathbb{M}_d^N$ and symmetric matrices $P_i \succ 0$, $i \in \mathcal{N}$, such that*

$$\Xi_{i,h} \prec 0,\ i \in \mathcal{N},\ h \in \{1,2\} \tag{23}$$

where $\Xi_{i,h} = \mathscr{E}(\tilde{A}_i, J_h, \sum_{j=1}^N \pi_{ji} P_j, P_i + (-1)^h \mu_h \tilde{Q}_i, T_s)$, $\tilde{Q}_i = e^{\tilde{A}_i^\top T_s} Q e^{\tilde{A}_i T_s}$.

(b) *There exist scalars $\mu_h > 0$, $h \in \{1,2\}$, a matrix $\Pi \in \mathbb{M}_d^N$ and a continuous symmetric matrix function $P_i(t) : [0, T_s] \to \mathbb{R}^{2n \times 2n}$, $i \in \mathcal{N}$, such that*

$$P_i(t) \succ 0,\ t \in [0, T_s],\ i \in \mathcal{N} \tag{24}$$

$$\mathscr{D}(\tilde{A}_i, P_i(t)) \prec 0,\ i \in \mathcal{N} \tag{25}$$

$$\Omega_{i,h} \prec 0,\ i \in \mathcal{N},\ h \in \{1,2\} \tag{26}$$

where $\Omega_{i,h} = J_h^\top \sum_{j=1}^N \pi_{ji} P_j(0) J_h - P_i(T_s) - (-1)^h \mu_h Q$.

(c) *There exist scalars $M \in \mathbb{N} \setminus \{0\}$, $\mu_h > 0$, $h \in \{1,2\}$, a matrix $\Pi \in \mathbb{M}_d^N$ and symmetric matrices $P_{i,m} \in \mathbb{R}^{2n \times 2n}$, $m \in \{0, \ldots, M\}$, $i \in \mathcal{N}$, such that, for $i \in \mathcal{N}$,*

$$P_{i,m} \succ 0,\ m \in \{0, \ldots, M\} \tag{27}$$

$$\mathscr{D}_1(\tilde{A}_i, P_{i,m+1}, P_{i,m}, \delta) \prec 0,\ m \in \{0, \ldots, M-1\} \tag{28}$$

$$\mathscr{D}_2(\tilde{A}_i, P_{i,m+1}, P_{i,m}, \delta) \prec 0,\ m \in \{0, \ldots, M-1\} \tag{29}$$

$$\Omega_{i,h} \prec 0,\ h \in \{1,2\} \tag{30}$$

where $\delta = T_s/M$ and $\Omega_{i,h} = J_h^\top \sum_{j=1}^N \pi_{ji} P_{j,0} J_h - P_{i,M} - (-1)^h \mu_h Q$.

when one of the above equivalent statements holds, then system (13)–(15) with $\omega(t) = 0$ is GUAS with switching signal (17) with P_i by statement (a), $P_i = P_i(0)$ by statement (b) and $P_i = P_{i,0}$ by statement (c), respectively.

Proof. The structure of the proof is as follows: First, we prove the equivalence by deriving (c) ⇒ (b) ⇒ (a) ⇒ (c), then establish GUAS by (a) ⇒ GUAS.

(c) ⇒ (b): Dividing interval $\mathcal{I} \triangleq [0, T_s]$ can into $M \in \mathbb{N} \setminus \{0\}$ segments described as $\mathcal{I}_m \triangleq [\delta_m, \delta_{m+1})$, $m = 0, 1, \ldots, M-1$, which are of equal length $\delta = T_s/M$, and then $\delta_0 = 0$ and $\delta_m = m\delta = \frac{mT_s}{M}$. Based on the discretization of $\mathcal{I}$, the following time-scheduled matrices $P_i(t)$, $i \in \mathcal{N}$, are introduced

$$\begin{cases} P_i(t) = (1 - \theta(t)) P_{i,m} + \theta(t) P_{i,m+1} \\ \theta(t) = Mt/T_s - m \end{cases},\ t \in \mathcal{I}_m \tag{31}$$

by which it can be seen that $0 \leq \theta(t) \leq 1$ and $P_i(t)$ defines a piecewise linear matrix function over $\mathcal{I}$.

By the definition of $P_i(t)$, $i \in \mathcal{N}$, as (31), we have $P_i(0) = P_{i,0}$ and $P_i(T_s) = P_{i,M}$, so (27) and (30) can make sure that (24) and (26) hold.

Then, one has

$$\mathcal{D}^+ P_i(t) = (P_{i,m+1} - P_{i,m}) \mathcal{D}^+ \theta(t),\ t \in \mathcal{I}_m \tag{32}$$

Due to $\theta(t) = M(t - \delta_m)/T_s$, we have $\mathcal{D}^+ \theta(t) = M/T_s$. Hence $\mathcal{D}^+ P_i(t)$ becomes

$$\mathcal{D}^+ P_i(t) = M(P_{i,m+1} - P_{i,m})/T_s,\ t \in \mathcal{I}_m \tag{33}$$

Thus, (28) and (29) imply (25) holds.

(b) $\Rightarrow$ (a): Pre- and post-multiplying (25) with $e^{\tilde{A}_i^\top t}$ and its transpose, and integrate it over $[0, T_s]$, it arrives

$$e^{\tilde{A}_i^\top T_s} P_i(T_s) e^{\tilde{A}_i T_s} - P_i(0) \prec 0,\ i \in \mathcal{N} \tag{34}$$

which implies $P_i(0) \succ e^{\tilde{A}_i^\top T_s} P_i(T_s) e^{\tilde{A}_i T_s}$, $i \in \mathcal{N}$. Furthermore, it equals to

$$P_i(T_s) \prec e^{-\tilde{A}_i^\top T_s} P_i(0) e^{-\tilde{A}_i T_s},\ i \in \mathcal{N} \tag{35}$$

Using (35) into (26), the following inequality holds for $i \in \mathcal{N}$ and $h \in \{1, 2\}$,

$$J_h^\top \sum\nolimits_{j=1}^N \pi_{ji} P_j(0) J_h - e^{-\tilde{A}_i^\top T_s} P_i(0) e^{-\tilde{A}_i T_s} - (-1)^h \mu_h Q \prec 0 \tag{36}$$

Letting $P_i = P_i(0) \succ 0$, $i \in \mathcal{N}$, (36) equals to

$$e^{\tilde{A}_i^\top T_s} J_h^\top \sum\nolimits_{j=1}^N \pi_{ji} P_j J_h e^{\tilde{A}_i T_s} - P_i - (-1)^h \mu_h \tilde{Q}_i \prec 0 \tag{37}$$

where $\tilde{Q}_i = e^{\tilde{A}_i^\top T_s} Q e^{\tilde{A}_i T_s}$. Thus, (23) can be established by letting $P_i = P_i(0) \succ 0$, $i \in \mathcal{N}$.

(a) $\Rightarrow$ (c): Since (23) holds, it implies that the following inequality holds

$$J_h^\top \sum\nolimits_{j=1}^N \pi_{ji} P_j J_h - e^{-\tilde{A}_i^\top T_s} P_i e^{-\tilde{A}_i T_s} - (-1)^h \mu_h Q \prec 0 \tag{38}$$

which implies that there exists an $\epsilon^* > 0$ such that

$$J_h^\top \sum\nolimits_{j=1}^N \pi_{ji} P_j J_h - e^{-\tilde{A}_i^\top T_s} P_i e^{-\tilde{A}_i T_s} - (-1)^h \mu_h Q \prec -\epsilon^* I \tag{39}$$

Then, for any $\epsilon > 0$, we can let $P_{i,0} = \epsilon P_i / \epsilon^* \succ 0$, $i \in \mathcal{N}$ (This choice of $P_{i,0}$, $i \in \mathcal{N}$, maintains the same switching law generated by P_i, $i \in \mathcal{N}$.), and $\hat{\mu}_h = \epsilon \mu_h / \epsilon^* > 0$, $h \in \{1, 2\}$, such that

$$J_h^\top \sum\nolimits_{j=1}^N \pi_{ji} P_{j,0} J_h - e^{-\tilde{A}_i^\top T_s} P_{i,0} e^{-\tilde{A}_i T_s} - (-1)^h \hat{\mu}_h Q \prec -\epsilon I \tag{40}$$

Using Lemma 1, there always exists a sufficiently large M^* such that (27), (28), (29) always hold with $P_{i,m}$, $m \in \{0, \ldots, M\}$, $M > M^*$, $i \in \mathcal{N}$, in the form of

$$P_{i,m} = e^{-\tilde{A}_i^\top \delta_m} P_{i,0} e^{-\tilde{A}_i \delta_m} - Z_{i,m},\ m \in \{0, \ldots, M\} \tag{41}$$

where

$$Z_{i,m} = \int_0^{\delta_m} e^{-\tilde{A}_i^\top (\delta_m - t)} Y_i(t) e^{-\tilde{A}_i (\delta_m - t)(t)} \mathrm{d}t$$

with $\delta_m = m T_s / M$, $m = \{0, \ldots, M\}$, and continuous matrix functions $Y_i(t) \succ 0$, $i \in \mathcal{N}$.

Thus, it yields

$$P_{i,M} = e^{-\tilde{A}_i^\top T_s} P_{i,0} e^{-\tilde{A}_i T_s} - Z_{i,M} \tag{42}$$

Substituting $e^{-\tilde{A}_i^\top T_s} P_{i,0} e^{-\tilde{A}_i T_s} = P_{i,M} + Z_{i,M}$ into (40), we have

$$J_h^\top \sum\nolimits_{j=1}^N \pi_{ji} P_{j,0} J_h - P_{i,M} - (-1)^h \hat{\mu}_h Q \prec -\epsilon I + Z_{i,M} \tag{43}$$

Since $\epsilon > 0$ can be arbitrarily chosen, we can choose a sufficiently large $\epsilon > 0$ such that

$$J_h^\top \sum\nolimits_{j=1}^N \pi_{ji} P_{j,0} J_h - P_{i,M} - (-1)^h \hat{\mu}_h Q \prec 0 \tag{44}$$

which implies that (30) holds.

(a) $\Rightarrow$ **GUAS:** First, we consider the system state $\tilde{x}(t_k^+)$ at sampling instants, we have

$$\tilde{x}(t_{k+1}^+) = \begin{cases} J_1 e^{\tilde{A}_{\sigma(t_k^+)}T_s}\tilde{x}(t_k^+), & \tilde{x}^\top(t_{k+1}^-)Q\tilde{x}(t_{k+1}^-) > 0 \\ J_2 e^{\tilde{A}_{\sigma(t_k^+)}T_s}\tilde{x}(t_k^+), & \tilde{x}^\top(t_{k+1}^-)Q\tilde{x}(t_{k+1}^-) \le 0 \end{cases} \tag{45}$$

Due to $\tilde{x}(t_{k+1}^-) = e^{\tilde{A}_{\sigma(t_k^+)}T_s}\tilde{x}(t_k^+)$, and letting $\tilde{Q}_i = e^{\tilde{A}_i^\top T_s}Qe^{\tilde{A}_i T_s}$, $k = t_k^+$, $\tilde{x}(k)$ evolves according to the following dynamics

$$\tilde{x}(k+1) = \begin{cases} J_1 e^{\tilde{A}_{\sigma(k)}T_s}\tilde{x}(k), & \tilde{x}^\top(k)\tilde{Q}_i\tilde{x}(k) > 0 \\ J_2 e^{\tilde{A}_{\sigma(k)}T_s}\tilde{x}(k), & \tilde{x}^\top(k)\tilde{Q}_i\tilde{x}(k) \le 0 \end{cases} \tag{46}$$

where $\sigma(k) = \arg\min_{i\in\mathcal{N}} \tilde{x}^\top(k)P_i\tilde{x}(k)$ and P_i, $i \in \mathcal{N}$, is same as in switching signal (17).

Construct Lyapunov function candidate as $V(\tilde{x}(k)) = \tilde{x}^\top(k)P_{\sigma(k)}\tilde{x}(k)$ and define $\Delta V(\tilde{x}(k)) = V(\tilde{x}(k+1)) - V(\tilde{x}(k))$, under the min-switching law (17), we have

$$\begin{aligned} \Delta V(\tilde{x}(k)) &= \min_{j\in\mathcal{N}} \tilde{x}^\top(k+1)P_j\tilde{x}(k+1) - \tilde{x}^\top(k)P_i\tilde{x}(k) \\ &\le \tilde{x}^\top(k+1)\left(\sum\nolimits_{j=1}^{N}\pi_{ji}P_j\right)\tilde{x}(k+1) - \tilde{x}^\top(k)P_i\tilde{x}(k) \end{aligned}$$

By (46), $\Delta V(\tilde{x}(k))$ arrives

$$\Delta V(\tilde{x}(k)) = \begin{cases} \tilde{x}^\top(k)\Gamma_{i,1}\tilde{x}(k), & \tilde{x}^\top(k)\tilde{Q}_i\tilde{x}(k) > 0 \\ \tilde{x}^\top(k)\Gamma_{i,2}\tilde{x}(k), & \tilde{x}^\top(k)\tilde{Q}_i\tilde{x}(k) \le 0 \end{cases} \tag{47}$$

where $\Gamma_{i,1} = \mathscr{E}(\tilde{A}_i, J_1, \sum_{j=1}^N \pi_{ji}P_{j,0}, P_i, T_s)$, $\Gamma_{i,2} = \mathscr{E}(\tilde{A}_i, J_2, \sum_{j=1}^N \pi_{ji}P_{j,0}, P_i, T_s)$. Since (23) holds, it implies there exists a sufficiently small $\epsilon > 0$ such that $\Xi_{i,h} < -\epsilon I$, $\forall i \in \mathcal{N}$, $h \in \{1, 2\}$ then using *S-Procedure*, it ensures that

$$\Delta V(\tilde{x}(k)) < -\epsilon \left\|\tilde{x}(k)\right\|^2,\ k \in \mathbb{N} \tag{48}$$

Letting $\lambda_{\min}$, $\lambda_{\max}$ be the minimal and maximal eigenvalues of P_i, $i \in \mathcal{N}$, respectively, it implies that $\lambda_{\min}\left\|\tilde{x}(k)\right\|^2 \le V(\tilde{x}(k)) \le \lambda_{\max}\left\|\tilde{x}(k)\right\|^2$. Thus, (48) implies that $V(\tilde{x}(k)) < (1 - \epsilon/\lambda_{\max})^k V(\tilde{x}(t_0))$, where $0 < 1 - \epsilon/\lambda_{\max} < 1$. Due to $k = t_k/T_s$, one has

$$V(\tilde{x}(t_k^+)) < e^{(t_k - t_0)\ln\frac{1-\epsilon/\lambda_{\max}}{T_s}}V(\tilde{x}(t_0)),\ t_k \in \mathcal{S} \tag{49}$$

Furthermore, it arrives

$$\left\|\tilde{x}(t_k^+)\right\| < \sqrt{\frac{\lambda_{\max}}{\lambda_{\min}}}e^{-\rho(t_k - t_0)}\left\|\tilde{x}(t_0)\right\|,\ t_k \in \mathcal{S} \tag{50}$$

where $\rho = -\ln(1 - \epsilon/\lambda_{\max})/2T_s > 0$.

Then, let us consider any $t \in (t_k, t_{k+1})$, the dynamics of mode i yields $\tilde{x}(t) = e^{\tilde{A}_i(t-t_k)}\tilde{x}(t_k^+)$, $t \in (t_k, t_{k+1})$. Using the following derivation

$$\left\|e^{\tilde{A}_i(t-t_k)}\right\| \le e^{\left\|\tilde{A}_i(t-t_k)\right\|} \le e^{\left\|\tilde{A}_i\right\|T_s},\ t \in (t_k, t_{k+1}) \tag{51}$$

we have $\left\|\tilde{x}(t)\right\| \le c\left\|\tilde{x}(t_k)\right\|$, $t \in (t_k, t_{k+1})$, where $c = \max_{i\in\mathcal{N}} e^{\left\|\tilde{A}_i\right\|T_s}$. Thus, by (50), it can be obtained that $\left\|\tilde{x}(t)\right\| < Ce^{-\rho(t-t_0)}\left\|\tilde{x}(t_0)\right\|$, where $C = ce^{\rho T_s}\sqrt{\lambda_{\max}/\lambda_{\min}} > 0$, and the GUAS can be established by the existence of $\mathcal{KL}$ function $\beta(\left\|\tilde{x}(t_0)\right\|, t) = Ce^{-\rho(t-t_0)}\left\|\tilde{x}(t_0)\right\|$. □

Some observations are obtained for three conditions in Theorem 1:

1. If no event-triggering condition is considered and the state $x(t)$ updates at each sampling instant, event-triggered system (13)–(15) is reduced to (7)–(9), and as a result, (23) can be rewritten to

$$\mathscr{E}(\tilde{A}_i, J, \sum\nolimits_{j=1}^{N} \pi_{ji} P_j, P_i, T_s) \prec 0,\ \ i \in \mathcal{N} \tag{52}$$

   It can be found that (52) recovers the result in [32], which deals with the switched system with min-switching law (16) only acts at sampling instant t_k. Theorem 1 generalizes the sampled switching case to event-triggered switching case. Furthermore, if we consider the *passive* switching, which means switched system could switch to any subsystems at every switching instant t_k. That means, for any $j \neq i,\ i, j \in \mathcal{N}$, we have to let $\pi_{ji} = 1$ and $\pi_{pi} = 0$, $p \neq j$, so

$$\mathscr{E}(\tilde{A}_i, J, P_j, P_i, T_s) \prec 0,\ \ i, j \in \mathcal{N} \tag{53}$$

   which exactly recovers the result in [23].

   The basic idea of Condition (a) is to consider the evolution of system state at sampling instant t_k, and the asymptotic convergence of $\tilde{x}(t_k)$ guarantees the asymptotic stability of system (13)–(15). However, if one attempts to make some further extensions of Condition (a) such as robust sampling case and $\mathcal{L}_2$-gain performance analysis, the presence of exponential term $e^{\tilde{A}_i T_s}$ makes such extensions difficult.

2. Condition (b) basically is an extension of the result in [28], from dwell time switching to periodically event-triggered switching. Regardless of event-triggering condition, system (7)–(9) is a switched system with a periodic dwell time T_s, and if we deactivate the switching rule (17) to consider *passive* switching, it leads to $\pi_{ji} = 1$ and $\pi_{pi} = 0$, $p \neq j$, thus (26) is rewritten to

$$P_j(0) - P_i(T_s) \prec 0,\ \ j \neq i,\ \ i, j \in \mathcal{N} \tag{54}$$

   Together with (24), (25), the result in [28] is recovered.

   Still consider system (7)–(9) regardless of event-triggering condition, (26) becomes

$$\sum\nolimits_{j=1}^{N} \pi_{ji} P_j(0) - P_i(T_s) \prec 0 \tag{55}$$

   Then, let us consider the special case with sampling interval $T_s \to 0$. In this case, we have to let the continuous matrix function $P_i(t) = P_i$, $i \in \mathcal{N}$, then (25) implies $\mathscr{D}(\tilde{A}_i, P_i) = \mathscr{C}(\tilde{A}_i, P_i)$, and (26) arrives at

$$\sum\nolimits_{j=1}^{N} \pi_{ji} P_j - P_i \prec 0 \tag{56}$$

   From the fact of $\sum_{j=1}^{N} \pi_{ji} P_j - P_i = \sum_{j=1}^{N} \phi_{ji} P_j$, $\Phi \in \mathbb{M}_c^N$, $i, j \in \mathcal{N}$, (56) leads to

$$\sum\nolimits_{j=1}^{N} \phi_{ji} P_j \prec 0 \tag{57}$$

   Combining (25), (57), the following result can be established

$$\mathscr{C}(\tilde{A}_i, P_i) + \sum\nolimits_{j=1}^{N} \phi_{ji} P_j \prec 0,\ \ \Phi \in \mathbb{M}_c^N,\ \ i, j \in \mathcal{N} \tag{58}$$

   which exactly recovers result in [23] for min-switching rule. Therefore, Condition (b) is an extension to sampling case and further to event-triggered case. One point need to be noted this

Table 1: Computational complexities of Conditions (a), (b) and (c) with fixed $\Pi \in \mathbb{M}_d^N$

	Number of Variables	LMI Constraints
Condition (a)	$(4n^2N + 2nN)/2 + 2$	$6nN + 2$
Condition (b)	∞	∞
Condition (c)	$(4n^2 + 2n)(M + 1)/2 + 2$	$6nN(M + 1) + 2$

min-switching law may introduce Zeno behaviors, but if we let T_s be a positive constant in our periodic event-triggered rule, one advantage is the elimination of Zeno behavior in switching.

In comparison with Condition (a), Condition (b) does not have any exponential terms which facilitates its further extensions to solve other problems. However, it is not numerically testable to check the existence of such time-varying matrix functions $P_i(t)$, $i \in \mathcal{N}$.

3. Condition (c) is a discretized version Condition (b), and similar as what has been discuss for Condition (b), if we discard the event-triggering condition and (30) becomes

$$\sum\nolimits_{j=1}^{N} \pi_{ji} P_{j,0} - P_{i,M} \prec 0 \tag{59}$$

which recovers the result in [22]. Moreover, if we further deactivate the min-switching strategy, (30) can be reduced to

$$P_{j,0} - P_{i,M} \prec 0 \tag{60}$$

to recover the result in [24] for switched system under dwell time constraint.

With a particularly constructed $P_i(t)$, $i \in \mathcal{N}$, Condition (c) recasts the search for a continuous matrix function $P_i(t)$ as a finite number of matrices $P_{i,m}$, $m \in \{0, \dots, M\}$, $i \in \mathcal{N}$, which is solvable for many current tools.

4. Though the three conditions are equivalent, the computation complexities are different. Condition (a) looks simpler and computationally much more efficient, see Table 1 for the comparison of computational complexities with a prescribed $\Pi \in \mathbb{M}_d^N$. Condition (b) is actually not numerically tractable by the present tools, so a special structure of $P_i(t)$, $i \in \mathcal{N}$, is employed in Condition (c), it turns the infinite number of decision variables in time-varying $P_i(t)$, $i \in \mathcal{N}$ into a finite number of matrices $P_{i,m}$, $m \in \{0, \dots, M\}$, $i \in \mathcal{N}$. However, the equivalency of Condition (c) to Conditions (a) and (b) has to be established based on a sufficient large M, and the computation cost increases as M grows, see Table 1. Though more computation cost has to pay in Condition (c), the further extensions beyond stability become possible.

Example 1 *Consider a switched system with two modes*

$$\left[\begin{array}{c} A_1 \\ \hdashline B_1^\top \end{array}\right] = \left[\begin{array}{cc} 1 & 3 \\ 6 & -2 \\ \hdashline 1 & 0.5 \end{array}\right], \quad \left[\begin{array}{c} A_2 \\ \hdashline B_2^\top \end{array}\right] = \left[\begin{array}{cc} -1.3 & -1.6 \\ -3.3 & 0.3 \\ \hdashline 0.2 & 0.3 \end{array}\right]$$

The feedback gains are $K_1 = [-5.1744 \quad -5.1904]$ and $K_2 = [18.7593 \quad 16.3442]$, which ensure the $A_i + B_iK_i$, $i \in \{1, 2\}$, are Hurwitz stable. The event triggering condition is $\Gamma(x(t_k), \hat{x}(t_k)) = \|\hat{x}(t_k) - x(t_k)\| - \Delta\|x(t_k)\|$, where $\Delta > 0$. To search for $\Pi \in \mathbb{M}_d^N$, we define $\pi_{11} \in [0, 1]$ and $\pi_{12} \in [0, 1]$, then $\pi_{21} = 1 - \pi_{11}$ and $\pi_{22} = 1 - \pi_{12}$, respectively. The increments $d\pi_{11} = 0.1$ and

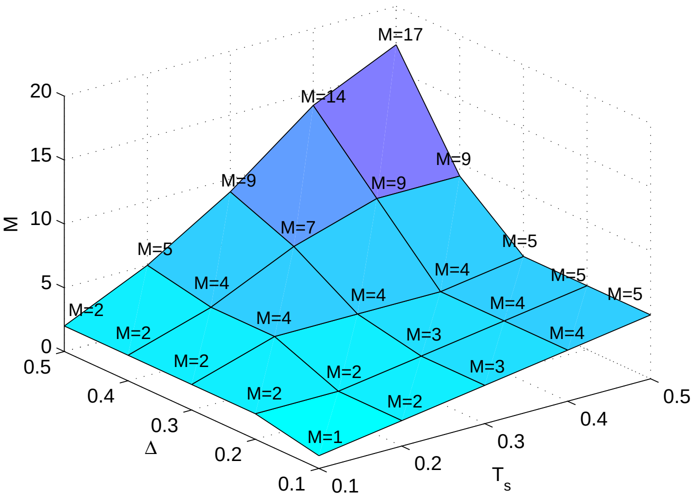


Figure 2: The least values of sufficiently large parameter M for Condition (c) to verify GUAS

Table 2: Computational time (second) of Condition (c) with a fixed $\Pi \in \mathbb{M}_d^N$

	$T_s = 0.1$	$T_s = 0.2$	$T_s = 0.3$	$T_s = 0.4$	$T_s = 0.5$
$\Delta = 0.1$	3.045	4.842	6.235	7.682	12.372
$\Delta = 0.2$	3.767	4.881	6.349	8.628	13.680
$\Delta = 0.3$	3.624	6.349	8.932	9.158	14.046
$\Delta = 0.4$	3.814	5.817	9.434	16.745	16.750
$\Delta = 0.5$	3.983	9.738	12.186	20.909	29.081

$d\pi_{12} = 0.1$ are taken to divide $[0, 1]$, and use the discretized points to turn the conditions in Conditions (a) and (c) into LMI feasibility problems.

First, we use Condition (a) to verify that the GUAS can be established with sampling times $T_s = \{0.1,\ 0.2,\ 0.3,\ 0.4,\ 0.5\}$ and state error $\Delta = \{0.1,\ 0.2,\ 0.3,\ 0.4,\ 0.5\}$. Then, to show the equivalence, we use Condition (c) to obtain same GUAS results, provided with sufficiently large parameters M. The results are shown in Figure 2.

Figure 2 shows the existence of sufficiently large M ensuring the equivalence of Conditions (a) and (c). However, the computational complexities of two theorems are different. The computational complexity of Condition (a) is fixed if the number of subsystems and system order are fixed, as Table 1 shows, but the computational complexity of Condition (c) increases as M grows. The computational time is given in Table 2. Larger Δ or T_s will lead to more computational time which is listed in Table 2 is because larger Δ or T_s needs larger M to establish the stability, as what Figure 1 shows. Taking the $T_s = 0.2$ for example, $\Delta = 0.2$ needs $M = 2$ and, on the other hand, $\Delta = 0.3$ needs $M = 4$. Larger M has more computational complexities as shown in Table 1. If the M are same, e.g. the case $T_s = 0.1$, $\Delta = 0.2$ and $\Delta = 0.3$ both need $M = 2$, so the computational times are similar.

Despite the equivalence of Conditions (a), (b) and (c), the main advantage of Condition (c) lies in its convenience of extending to solve further problems. In next sections, extensions will be made to robust sampling case and $\mathcal{L}_2$-gain performance analysis for event-triggered switched control system based on Condition (c).

4 Robust Sampling Scheme

In this section, the uncertainties in sampling interval will be considered. To further develop a robust switching rule (17), the sampling interval is generalized to $T_s \in [T_{\min}, T_{\max}]$. Similar as the generalization from periodic switching to aperiodic switching in [30], the generalization of Conditions (a) and (b) in Theorem 1 can be made simply by replace a fixed T_s by a variable $\tau \in [T_{\min}, T_{\max}]$ in these conditions. For instance, Condition (a) can be directly generalized as

$$\mathscr{E}(\tilde{A}_i, J_h, \sum\nolimits_{j=1}^{N} \pi_{ji} P_j, P_i + (-1)^h \mu_h e^{\tilde{A}_i^\top \tau} Q e^{\tilde{A}_i \tau}, \tau) \prec 0,\ i \in \mathcal{N},\ h \in \{1,2\} \tag{61}$$

holds for all $\tau \in [T_{\min}, T_{\max}]$. However, it is difficult to check (61) for all $\tau \in [T_{\min}, T_{\max}]$ which has infinitely many number for checking in an interval $[T_{\min}, T_{\max}]$, due to the continuity argument and intricate dependence of (61) with $\tau \in [T_{\min}, T_{\max}]$. Thus, it is difficult to numerically verify the stability by (61) which actually requires infinite values for checking.

In order to establish a numerically tractable method for robust sampling interval $T_s \in [T_{\min}, T_{\max}]$, we resort to generalize Condition (c). Like the extension from dwell time to ranged dwell time in [30] for sampled-data systems, the following theorem can be developed for robust sampling interval in the framework of event-triggered control scheme.

Theorem 2 *Consider event-triggered switched control system (13)–(15) with $\omega(t) = 0$, if there exist scalars $M \in \mathbb{N} \setminus \{0\}$, $\mu_h > 0$, $h \in \{1,2\}$, a matrix $\Pi \in \mathbb{M}_d^N$ and symmetric matrices $P_{i,m} \in \mathbb{R}^{2n\times 2n}$, $m \in \{0,\dots,M\}$, $i \in \mathcal{N}$, such that, for $i \in \mathcal{N}$,*

$$P_{i,m} \succ 0,\ m \in \{0,\dots,M\} \tag{62}$$

$$\mathscr{D}_1(\tilde{A}_i, P_{i,m+1}, P_{i,m}, \delta) \prec 0,\ m \in \{0,\dots,M-1\} \tag{63}$$

$$\mathscr{D}_2(\tilde{A}_i, P_{i,m+1}, P_{i,m}, \delta) \prec 0,\ m \in \{0,\dots,M-1\} \tag{64}$$

$$\Omega_{i,h,\hat{m}} \prec 0,\ \hat{m} \in \{\underline{M},\dots,M\},\ h \in \{1,2\} \tag{65}$$

where $\delta = T_{\max}/M$, $\underline{M} = \mathrm{int}\{\frac{MT_{\min}}{T_{\max}}\}$ and $\Omega_{i,h,\hat{m}} = J_h^\top \sum_{j=1}^{N} \pi_{ji} P_{j,0} J_h - P_{i,\hat{m}} - (-1)^h \mu_h Q$, then system (13)–(15) with $\omega(t) = 0$ is GUAS under sampled switching rule (17) with $P_i = P_{i,0}$, $i \in \mathcal{N}$.

Proof. Since $\underline{M} = \mathrm{int}\{\frac{MT_{\min}}{T_{\max}}\}$, we have $\frac{\underline{M}T_{\max}}{M} \le T_{\min}$ which implies that the interval $[T_{\min}, T_{\max}] \subseteq \bigcup_{\hat{m}=\underline{M},\dots,M-1} \mathcal{I}_{\hat{m}}$.

Considering $P_i(t)$, $t \in [0, T_{\max}]$ defined by

$$\begin{cases} P_i(t) = (1-\theta(t))P_{i,m} + \theta(t)P_{i,m+1} \\ \theta(t) = Mt/T_{\max} - m \end{cases},\ t \in \mathcal{I}_m \tag{66}$$

where $0 \le \theta(t) \le 1$. First by (62), we can obtain $P_i(t) \succ 0$, $t \in [T_{\min}, T_{\max}]$. Then, (63) and (64) have $\mathcal{D}(\tilde{A}_i, P_i(t)) \prec 0$, and for any $\tau \in [T_{\min}, T_{\max}]$, it is obtained

$$P_{i,0} = P_i(0) \succ e^{\tilde{A}_i^\top \tau} P_i(\tau) e^{\tilde{A}_i \tau},\ \tau \in [T_{\min}, T_{\max}] \tag{67}$$

by integrating $\mathcal{D}(\tilde{A}_i, P_i(t)) \prec 0$ over $[0, \tau]$. Then, (65) implies that

$$e^{\tilde{A}_i^\top \tau} J_h^\top \sum\nolimits_{j=1}^{N} \pi_{ji} P_{j,0} J_h e^{\tilde{A}_i^\top \tau} - e^{\tilde{A}_i^\top \tau} P_i(\tau) e^{\tilde{A}_i^\top \tau} - (-1)^h \mu_h \tilde{Q}(\tau) \prec 0 \tag{68}$$

holds for $\tau \in [T_{\min}, T_{\max}]$, where $\tilde{Q}_i(\tau) = e^{\tilde{A}_i^\top \tau} Q e^{\tilde{A}_i \tau}$. Using (67) into (68) and letting $P_i = P_{i,0}$, $i \in \mathcal{N}$, it reaches that

$$\mathscr{E}(\tilde{A}_i, J_h, \sum\nolimits_{j=1}^{N} \pi_{ji} P_j, P_i + (-1)^h \mu_h \tilde{Q}(\tau), \tau) \prec 0, \ \tau \in [T_{\min}, T_{\max}] \tag{69}$$

which is exactly (61), thus the robust GUAS can be established. □

In comparison with (61), the extension of Condition (a), which has an infinite many decision variables to search, Theorem 2 only has a finite number of decision variable to check the GUAS for event-triggered switched system with ranged sampling intervals. The numerically tractable feature is an obvious advantage over (61) which is a straightforward extension from Condition (a), and this promising feature of Theorem 2 which is actually a generalization of Condition (c) basically benefits from the fact that the system matrices $\tilde{A}_i$ are affine in the corresponding conditions.

5 $\mathcal{L}_2$-Gain Performance Analysis

In the presence of disturbance $\omega(t)$, $\mathcal{L}_2$-gain performance is a disturbance attenuation performance for event-triggered switched system (13)–(15). The basic idea of Condition (a) in Theorem 1, that is abstracting continuous-time system (13)–(15) into a discrete-time version, is difficult to be extended from stability analysis to $\mathcal{L}_2$-gain performance analysis, since the discrete-time abstraction only defines the input-output relation at sampling instants t_k, losing the information over interval (t_k, t_{k+1}). Moreover, the technical difficulties for extension mainly lies in the exponential term $e^{\tilde{A}_i T_s}$. On the other hand, Condition (c) in Theorem 1 can be extended owing to the affineness in system matrix $\tilde{A}_i$. In the following, a numerically tractable result is proposed for $\mathcal{L}_2$-gain performance analysis.

Theorem 3 *Consider event-triggered switched control system (13)–(15), if there exist scalars $M \in \mathbb{N}\backslash\{0\}$, $\mu_h > 0$, $h \in \{1, 2\}$, a matrix $\Pi \in \mathbb{M}_d^N$ and symmetric matrices $P_{i,m} \in \mathbb{R}^{2n\times 2n}$, $m \in \{0, \ldots, M\}$, $i \in \mathcal{N}$, such that, for $i \in \mathcal{N}$,*

$$P_{i,m} \succ 0, \ m \in \{0, \ldots, M\} \tag{70}$$

$$\Xi_{i,m,1} \prec 0, \ m \in \{0, \ldots, M-1\} \tag{71}$$

$$\Xi_{i,m,2} \prec 0, \ m \in \{0, \ldots, M-1\} \tag{72}$$

$$\Omega_{i,h} \prec 0, \ h \in \{1, 2\} \tag{73}$$

where $\Omega_{i,h} = J_h^\top \sum_{j=1}^{N} \pi_{ji} P_{j,0} J_h - P_{i,M} - (-1)^h \mu_h Q$, and

$$\Xi_{i,m,1} = \begin{bmatrix} \mathscr{D}_1(\tilde{A}_i, P_{i,m+1}, P_{i,m}, T_s/M) & * & * \\ \tilde{E}_i^\top P_{i,m+1} & -\gamma^2 I & * \\ \tilde{C}_i & \tilde{D}_i & -I \end{bmatrix}$$

$$\Xi_{i,m,2} = \begin{bmatrix} \mathscr{D}_2(\tilde{A}_i, P_{i,m+1}, P_{i,m}, T_s/M) & * & * \\ \tilde{E}_i^\top P_{i,m} & -\gamma^2 I & * \\ \tilde{C}_i & \tilde{D}_i & -I \end{bmatrix}$$

then switched system (13)–(15) is GUAS and has an $\mathcal{L}_2$-gain γ under sampled switching rule (17) with $P_i = P_{i,0}$, $i \in \mathcal{N}$.

Proof. The GUAS can be easily obtained by Condition (c) in Theorem 1, thus we focus on the $\mathcal{L}_2$-gain performance in the following. First, we let

$$\Omega(t) = \|y(t)\|^2 - \gamma^2 \|\omega(t)\|^2 \tag{74}$$

and

$$J_k(t) = \int_{t_k^+}^{t} \Omega(s)\mathrm{d}s, \;\; t \in [t_k^+, t_{k+1}^-] \tag{75}$$

which can imply that

$$J_k(t^-) = \int_{t_k^+}^{t^-} \left(\Omega(s) + \mathcal{D}^+ V_i(\tilde{x}(s))\right) \mathrm{d}s - V_i(\tilde{x}(t^-)) + V_i(\tilde{x}(t_k^+)) \tag{76}$$

where $V_i(\tilde{x}(t))$ is defined as $V_i(\tilde{x}(t)) = \tilde{x}^\top(t)P_i(t)\tilde{x}(t)$, $i \in \mathcal{N}$, with $P_i(t)$, $i \in \mathcal{N}$, defined by (31).

Then, by (75), it can be deduced that $\int_{t_0}^{\infty} \Omega(s)\mathrm{d}s = \sum_{k=0}^{\infty} J_k(t_{k+1}^-)$, which can be rewritten as

$$\int_{t_0}^{\infty} \Omega(s)\mathrm{d}s = \sum\nolimits_{k=0}^{\infty} \int_{t_k^+}^{t_{k+1}^-} \left(\Omega(s) + \mathcal{D}^+ V_i(\tilde{x}(s))\right) \mathrm{d}s + \sum\nolimits_{k=1}^{\infty} \left(V_j(\tilde{x}(t_k^+)) - V_i(\tilde{x}(t_k^-))\right) + V_i(\tilde{x}(t_0)) \tag{77}$$

From (73), one has

$$V_j(\tilde{x}(t_k^+)) - V_i(\tilde{x}(t_k^-)) \leq 0, \forall t_k \in \mathcal{S} \tag{78}$$

is satisfied with min-switching rule (17). Moreover, it is obtained that

$$\Omega(t) + \mathcal{D}^+ V_i(\tilde{x}(t)) = \zeta^\top(t) \begin{bmatrix} \Lambda_i & P_i(t)\tilde{E}_i + \tilde{C}_i^\top D_i \\ * & \tilde{D}_i^\top \tilde{D}_i - \gamma^2 I \end{bmatrix} \zeta(t) \tag{79}$$

where $\zeta^\top = [\tilde{x}^\top(t) \;\; \omega^\top(t)]$, $\Lambda_i = \mathscr{D}(\tilde{A}_i, P_i(t)) + \tilde{C}_i^\top \tilde{C}_i$.

Thus, from (71), (72), it gives $\Omega(t) + \mathcal{D}^+ V_i(\tilde{x}(t)) < 0$. Together with (78) and $\tilde{x}(t_0) = 0$, we obtain

$$\int_{t_0}^{\infty} \Omega(s)\mathrm{d}s < 0 \tag{80}$$

which leads to $\int_{t_0}^{\infty} \|y(t)\|^2 \mathrm{d}t \leq \gamma^2 \int_{t_0}^{\infty} \|\omega(t)\|^2 \mathrm{d}t$ when $\omega(t) \neq 0$. Therefore, the $\mathcal{L}_2$-gain performance is guaranteed. The proof is complete. □

From Theorem 3, it should be stressed that although the min-switching rule (17) only acts at sampling instants $t_k \in \mathcal{S}$, the $\mathcal{L}_2$-gain level which is defined over $[t_0, \infty)$ can be estimated. This is because (71) and (72) fully characterize the input-output property in the sense of $\mathcal{L}_2$-gain during $[t_k, t_{k+1})$. Moreover, if the robust sampling scheme is considered, the similar extension can be easily made as Theorem 2.

Under the framework of Theorem 3, an estimate of the $\mathcal{L}_2$-gain can be obtained by

$$\begin{gathered} \min \gamma^2 \\ \text{s.t. } (70),\; (71),\; (72),\; (73) \end{gathered} \tag{81}$$

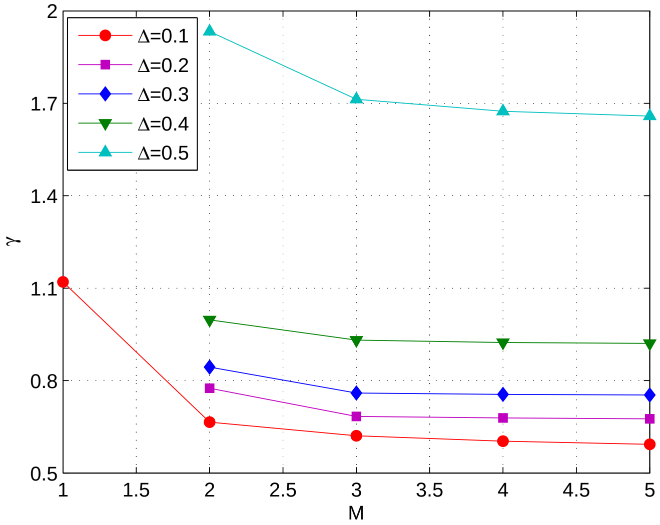


Figure 3: Suboptimal $\mathcal{L}_2$-gain γ with respect to different M

Same as the stability analysis result, the computational results obtained by solving the linear-matrix-inequality-based optimization problems (81) also depend on the choice of M. Less conservative results will be obtained with larger M, at the expense of higher computational cost, which will be shown by the following example.

Example 2 *Consider a switched system with two modes same as in Example 1, and C_i, D_i, E_i, $i \in \{1, 2\}$ are chosen as below:*

$$C_1 = C_2 = [1 \quad 1], \;\; E_1 = E_2 = \begin{bmatrix} 0.2 \\ 0.5 \end{bmatrix}, \;\; D_1 = D_2 = 0.5 \tag{82}$$

We still consider $\pi_{11} \in [0, 1]$ and $\pi_{12} \in [0, 1]$ with $\pi_{21} = 1 - \pi_{11}$ and $\pi_{22} = 1 - \pi_{12}$, respectively, the increments $\Delta\pi_{11} = 0.1$ and $\Delta\pi_{12} = 0.1$ are taken to divide $[0, 1]$, and search the optimal γ for these discretized points by Theorem 3. The suboptimal $\mathcal{L}_2$-gain is obtained as the minimal value of the optimal γ of all discretized points. Furthermore, given a constant sampling time $T_s = 100$ ms and $\Delta = \{0.1, \; 0.2, \; 0.3, \; 0.4, \; 0.5\}$, the suboptimal $\mathcal{L}_2$-gain γ with respect to different M are shown in Figure 3. From Figure 3, it can be observed that the estimated $\mathcal{L}_2$-gain γ decreases as M increases, this is because that a larger M implies a finer division of the sampling interval, and thus a less conservative result can be obtained. Moreover, it can be also found that the control performance becomes worse with a larger state error Δ in event trigger condition, this is consistent with the actual situation. The increasing computational complexities along with M is same as in Table 2, which is not presented here.

6 Conclusions

In this paper, the event-triggered control for switched linear systems has been studied. Three stability criteria are proposed to ensure asymptotic stability of switched system subject to min-switching rule which is only allowed to activate at sampling instants. It has been proved that the three stability criteria are equivalent. Then, taking advantages of one stability criterion with affineness in system matrices, extensions to robust sampling scheme and $\mathcal{L}_2$-gain analysis. In the future work, the controller design, switching rule design and event-triggering condition design should be taken into account based on the stability analysis results proposed in this paper.

7 Acknowledgment

The authors would like to thank the reviewers for their helpful comments and suggestions to improve our work. The material presented in this paper is based upon work supported by the National Science Foundation (NSF) under grant numbers CNS 1464311, EPCN 1509804, and SHF 1527398, the Air Force Research Laboratory (AFRL) through contract number FA8750-15-1-0105, and the Air Force Office of Scientific Research (AFOSR) under contract numbers FA9550-15-1-0258 and FA9550-16-1-0246.

A Proof of Lemma 1

First, given (22) and $0 \prec Y(t) \prec \epsilon I$, $t \in [0, T_s]$, with a sufficiently small $\epsilon > 0$, obviously we can obtain

$$P_m \succ e^{-A^\top \delta_m} P_0 e^{-A\delta_m} - \epsilon \int_0^{\delta_m} e^{-A^\top (\delta_m - t)} e^{-A(\delta_m - t)(t)} \mathrm{d}t \succ 0, \ m \in \{0, \dots, M\}$$

holds for any initial $P_0 \succ 0$.

Letting $Z_m = \int_0^{\delta_m} e^{-A^\top (\delta_m - t)} Y(t) e^{-A(\delta_m - t)(t)} \mathrm{d}t$ and substituting (22) into $\mathscr{D}_1(A, P_{m+1}, P_m, \delta)$ to get

$$\mathscr{D}_1(A, P_{m+1}, P_m, \delta) = \vartheta_{m,1}(\delta) + \vartheta_{m,2}(\delta) + \vartheta_{m,3}(\delta) \tag{83}$$

where $\delta = T_s/M$ and

$$\begin{aligned} \vartheta_{m,1}(\delta) &= e^{-A^\top \delta_m} \Omega(h) e^{-A\delta_m} \\ \Omega(\delta) &= \mathscr{C}(A, P_0) + \mathscr{E}(A, I, P_0/\delta, P_0/\delta, \delta) \\ \vartheta_{m,2}(\delta) &= -\mathscr{C}(A, Z_m) \\ \vartheta_{m,3}(\delta) &= (Z_m - Z_{m+1})/\delta \end{aligned}$$

Due to $\lim_{\delta \to 0^+} \sup \mathscr{E}(A, I, P_0/\delta, P_0/\delta, \delta) = -\mathscr{C}(A, P_0)$, therefore it yields that $\lim_{\delta \to 0^+} \sup \Omega(\delta) = 0$, which implies

$$\lim_{\delta \to 0^+} \sup \vartheta_{m,1}(\delta) = 0 \tag{84}$$

Moreover, due to $0 \le \delta_m \le T_s$, it implies that $e^{-A\delta_m}$ is bounded, $\vartheta_{m,1}(\delta)$ uniformly converges to zero.

In addition, it can be seen that

$$\lim_{\delta \to 0^+} \sup \vartheta_{m,3}(\delta) = -Y(\delta_m) + \mathscr{C}(A, Z_m) \tag{85}$$

which results in

$$\lim_{\delta \to 0^+} \sup(\vartheta_{m,2}(\delta) + \vartheta_{m,3}(\delta)) = -Y(\delta_m) \tag{86}$$

which implies that $\lim_{\delta \to 0^+} \sup(\vartheta_{i,m,2}(\delta) + \vartheta_{m,3}(\delta)) \prec 0$ due to $Y(t) \succ 0$, $t \in [0, T_s]$.

In conclusion, with the aid of (84) and (86), we have

$$\lim_{\delta \to 0^+} \sup \mathscr{D}_1(A, P_{m+1}, P_m, \delta) \prec 0 \tag{87}$$

so there exists a sufficiently small δ_1^* such that

$$\mathscr{D}_1(A, P_{m+1}, P_m, \delta) \prec 0 \tag{88}$$

holds for all $\delta < \delta_1^*$.

By a similar procedure as above, we can consider $\mathscr{D}_2(A, P_{m+1}, P_m, \delta)$ to obtain

$$\lim_{\delta \to 0^+} \sup \mathscr{D}_2(A, P_{m+1}, P_m, \delta) \prec 0 \tag{89}$$

and we can find a sufficiently small δ_2^* such that

$$\mathscr{D}_2(A, P_{m+1}, P_m, \delta) \prec 0 \tag{90}$$

holds for all $\delta < \delta_2^*$.

By setting $\delta^* = \min\{\delta_1^*, \delta_2^*\}$, we can conclude that there exists a sufficiently small δ^* such that (20) and (21) hold for any $\delta < \delta^*$. Due to $\delta = T_s/M$, it is equivalent to the existence of a sufficiently large M^* such that (20) and (21) hold for any $M > M^*$.

References

[1] R. DeCarlo, M. S. Branicky, S. Pettersson, and B. Lennartson. Perspectives and results on the stability and stabilizability of hybrid systems. *Proceedings of the IEEE*, 88(7):1069–1082, 2000.

[2] H. Lin and P. J. Antsaklis. Stability and stabilizability of switched linear systems: a survey of recent results. *IEEE Transactions on Automatic Control*, 54(2):308–322, 2009.

[3] D. Liberzon. *Switching in Systems and Control.* Springer Science & Business Media, 2012.

[4] P. Hauroigne, P. Riedinger, and C. Iung. Switched affine systems using sampled-data controllers: robust and guaranteed stabilization. *IEEE Transactions on Automatic Control*, 56(12):2929–2935, 2011.

[5] L. Zhang, H. Gao, and O. Kaynak. Network-induced constraints in networked control systems: a survey. *IEEE Transactions on Industrial Informatics*, 9(1):403–416, 2013.

[6] W.P.M.H. Heemels, M.C.F. Donkers, and A. Teel. Periodic event-triggered control for linear systems. *IEEE Transactions on Automatic Control*, 58(4):847–861, 2013.

[7] R. Shorten, F. Wirth, O. Mason, K. Wulff, and C. King. Stability criteria for switched and hybrid systems. *SIAM Review*, 49(4):545–592, 2007.

[8] W. Xiang and J. Xiao. Stabilization of switched continuous-time systems with all modes unstable via dwell time switching. *Automatica*, 50(3):940–945, 2014.

[9] G. Chesi, P. Colaneri, J. C. Geromel, R. Middleton, and R. Shorten. A nonconservative LMI condition for stability of switched systems with guaranteed dwell time. *IEEE Transactions on Automatic Control*, 57(5):1297–1302, 2012.

[10] A. S. Morse. Supervisory control of families of linear set-point controllers part I. exact matching. *IEEE Transactions on Automatic Control*, 41(10):1413–1431, 1996.

[11] J. P. Hespanha. Stability of switched systems with average dwell-time. In *Proceedings of the 38th IEEE Conference on Decision and Control, 1999*, volume 3, pages 2655–2660, Phoenix, AZ, 1999.

[12] L. Zhang, S. Zhuang, and P. Shi. Non-weighted quasi-time-dependent $\mathcal{H}_\infty$ filtering for switched linear systems with persistent dwell-time. *Automatica*, 54:201–209, 2015.

[13] P. Tabuada. Event-triggered real-time scheduling of stabilizing control tasks. *IEEE Transactions on Automatic Control*, 52(9):1680–1685, 2007.

[14] W.P.M.H. Heemels, J.H. Sandee, and P.P.J. Van Den Bosch. Analysis of event-driven controllers for linear systems. *International Journal of Control*, 81(4):571–590, 2008.

[15] X. Wang and M. Lemmon. Self-triggered feedback control systems with finite-gain stability. *IEEE Transactions on Automatic Control*, 54(3):452–467, 2009.

[16] T. Henningsson, E. Johannesson, and A. Cervin. Sporadic event-based control of first-order linear stochastic systems. *Automatica*, 44(11):2890–2895, 2008.

[17] D. Ding, Z. Wang, B. Shen, and H. Dong. Event-triggered distributed $\mathcal{H}_\infty$ state estimation with packet dropouts through sensor networks. *IET Control Theory & Applications*, 9(13):1948–1955, 2015.

[18] H. Yu and F. Hao. Periodic event-triggered state-feedback control for discrete-time linear systems. *Journal of the Franklin Institute*, 353(8):1809–1828, 2016.

[19] C. Peng and J. Zhang. Event-triggered output-feedback $\mathcal{H}_\infty$ control for networked control systems with time-varying sampling. *IET Control Theory & Applications*, 9(9):1384–1391, 2015.

[20] H. Yan, S. Yan, H. Zhang, and H. Shi. $\mathcal{L}_2$ control design of event-triggered networked control systems with quantizations. *Journal of the Franklin Institute*, 352(1):332–345, 2015.

[21] W.P.M.H. Heemels, K.H. Johansson, and P. Tabuada. An introduction to event-triggered and self-triggered control. In *2012 IEEE 51st Annual Conference on Decision and Control (CDC)*, pages 3270–3285, 2012.

[22] L. Allerhand and U. Shaked. Robust state-dependent switching of linear systems with dwell time. *IEEE Transactions on Automatic Control*, 58(4):994–1001, 2013.

[23] J. C. Geromel and P. Colaneri. Stability and stabilization of continuous-time switched linear systems. *SIAM Journal on Control and Optimization*, 45(5):1915–1930, 2006.

[24] L. Allerhand and U. Shaked. Robust stability and stabilization of linear switched systems with dwell time. *IEEE Transactions on Automatic Control*, 56(2):381–386, 2011.

[25] C. Briat and A. Seuret. Affine characterizations of minimal and mode-dependent dwell-times for uncertain linear switched systems. *IEEE Transactions on Automatic Control*, 58(5):1304–1310, 2013.

[26] S. Pettersson and B. Lennartson. Stabilization of hybrid systems using a min-projection strategy. In *Proceedings of the 2001 American Control Conference*, volume 1, pages 223–228, Arlington, VA, 2001.

[27] C. Duan and F. Wu. Analysis and control of switched linear systems via dwell-time min-switching. *Systems & Control Letters*, 70:8–16, 2014.

[28] C. Briat. Convex conditions for robust stabilization of uncertain switched systems with guaranteed minimum and mode-dependent dwell-time. *Systems & Control Letters*, 78:63–72, 2015.

[29] W. Xiang. On equivalence of two stability criteria for continuous-time switched systems with dwell time constraint. *Automatica*, 54:36–40, 2015.

[30] C. Briat. Convex conditions for robust stability analysis and stabilization of linear aperiodic impulsive and sampled-data systems under dwell-time constraints. *Automatica*, 49(11):3449–3457, 2013.

[31] K. Gu, J. Chen, and V. L. Kharitonov. *Stability of Time-Delay Systems*. Birkhäuser, Basel, 2003.

[32] G. S. Deaecto, M. Souza, and J. C. Geromel. Chattering free control of continuous-time switched linear systems. *IET Control Theory & Applications*, 8(5):348–354, 2014.

Output Reachable Set Estimation for Switched Linear Systems and Its Application in Safety Verification

Weiming Xiang, Hoang-Dung Tran, and Taylor T. Johnson

***Abstract*—This paper addresses the output reachable set estimation problem for continuous-time switched linear systems consisting of Hurwtiz stable subsystems. Based on a common Lyapunov function approach, the output reachable set is estimated by a union of bounding ellipsoids. Then, multiple Lyapunov functions with time-scheduled structure are employed to estimate the output reachable set for switched systems under dwell time constraint. Furthermore, the safety verification problem of uncertain switched systems is investigated based on the result of output reachable set estimation. First, a sufficient condition ensuring the existence of an approximate bisimulation relation between two switched linear systems with a prescribed precision is proposed. Then, the safety verification for an uncertain switched system can be performed through an alternative safety verification for a switched system with exact parameters. Numerical examples are provided to illustrate our results.**

***Index Terms*—Reachable set estimation, safety verification, switched system, uncertain system.**

I. Introduction

Switched systems are a typical class of hybrid systems, which consist of a family of subsystems described by continuous or discrete-time dynamics, and a switching law that specifies the active subsystem at each time instant. Due to the multi-modal feature, switched systems can efficiently model practical systems that are inherently multi-modal, i.e., several dynamical subsystem models are required to describe their behaviors. So far, the research on switched systems has attracted significant attention and an extensive literature is by now available, for example in stability and stabilization [1]–[5], controllability and reachability analysis [6], $\mathcal{H}_\infty$ control and filtering [7]–[9].

Reachable set estimation aims to derive a closed bounded set that constrains all the state trajectories generated by a dynamic system with a prescribed initial state set and an input set. As its further extension, the output reachable set estimation is to derive a closed bounded set containing the set of all outputs of a system. Reachable set estimation problem is not only of theoretical interest in robust control theory [10], but also closely related to practical engineering for the safety verification problems [11]. In some early work, the reachable set bounding was considered in the context of state estimation and it has later received a lot of attention in parameter estimation, see [12] and references therein. Recently, many researchers have been interested in employing ellipsoidal techniques based on Lyapunov function approaches to estimate the reachable sets for different classes of systems. In the framework of bounding ellipsoid, the quadratic Lyapunov function has played a fundamental role in the reachable set estimation problem, and it has been further developed to time-delay systems [13]–[16], singular systems [17], discrete-time switched systems under arbitrary switching [18] and periodic switching [19]. However, according to the best of the authors' knowledge, the reachable set estimation for continuous-time switched systems with dwell-time restriction has not been fully investigated, and it therefore motivates our study.

In this paper, the contributions are two folds. First, we study the output reachable set estimation problem for continuous-time switched linear systems consisting of Hurwitz stable subsystems. In the arbitrary switching case, an over approximation of output reachable set is obtained as a union of a collection of bounding ellipsoids centered around origin and moreover, a linear matrix inequality (LMI) based optimization problem is formulated to obtain the smallest estimated reachable set. These results are all derived in the framework of a common Lyapunov function shared across modes, however, it may yield overly conservative results, especially when some information of switching laws is available. Thus, with regard to a class of time-dependent switching signal under dwell time constraint, a time-scheduled multiple Lyapunov function approach is further employed and preciser estimation results can be achieved. In particular, it is worth mentioning that this time-scheduled multiple Lyapunov function approach covers the common Lyapunov function approach. In some papers, e.g., [20], [21], the finite-time boundedness is used for bounding state trajectories of a system, but it focuses on a finite-time interval other than all time along the system operation. Furthermore, the estimation from initial time to infinity is necessary for some problems such as the bisimulation and safety verification in the second contribution in this paper.

Based on the results for output reachable set estimation and inspired by approximate bisimulation relations in [22]–[24], a sufficient condition is derived to establish the existence of approximate bisimulation of two switched linear systems. Then, since the safety verification for uncertain systems is

The material presented in this paper is based upon work supported by the National Science Foundation (NSF) under grant numbers CNS 1464311, EPCN 1509804, and SHF 1527398, the Air Force Research Laboratory (AFRL) through contract number FA8750-15-1-0105, and the Air Force Office of Scientific Research (AFOSR) under contract numbers FA9550-15-1-0258 and FA9550-16-1-0246.

Authors are with the Department of Electrical Engineering and Computer Science, Vanderbilt University, Nashville, Tennessee 37212, USA. Emails: xiangwming@gmail.com; trhoangdung@gmail.com; taylor.johnson@gmail.com

difficult due to the uncertain time-varying coefficients in the system matrices, one would ask: *Can we find a bisimilar system with exact parameters for an uncertain system and perform a safety verification for the bisimilar system to ensure the safety of the uncertain system?* In this paper, an LMI-based method is proposed to convert the uncertain switched system into a switched system with exact parameters along with a precision between two systems, so that the safety verification for uncertain systems can be performed by verifying the safety of the transformed systems, avoiding the difficulties in handling the uncertainties.

The rest of this paper is organized as follows. Some preliminaries and problem formulation are given in Section II. The main results on output reachable set estimation is proposed in Section III. In Section IV, the application to safety verification for uncertain switched systems is presented. Conclusions are given in Section V.

Notation: $\mathbb{N}$ represents the set of natural numbers. $\mathbb{R}$ and $\mathbb{R}_{\geq 0}$ denote the fields of real numbers and nonnegative real numbers, respectively. $\mathbb{R}^n$ is the vector space of all n-tuples of real numbers, $\mathbb{R}^{n\times n}$ is the space of $n \times n$ matrices with real entries. $\mathbb{S}_{+}^{n\times n}$ is the set of real symmetric positive definite $n \times n$ matrices. The notation $P \succ 0$ ($P \prec 0$) means P is real symmetric and positive definite (negative definite). $A^\top$ denotes the transpose of A, and we let $\mathrm{Sym}(A) = A^\top + A$. In symmetric block matrices, we use * as an ellipsis for the terms that are introduced by symmetry. $\mathrm{diag}\{\cdots\}$ denotes a block-diagonal matrix. $\|\cdot\|$ stands for the Euclidean norm. The bounding ellipsoid is expressed by $\mathcal{E}(R) \triangleq \{x \in \mathbb{R}^n \mid x^\top Rx \leq 1, R \in \mathbb{S}_{+}^{n\times n}\}$, and ball $\mathcal{B}(x_0, \delta) \triangleq \{x \in \mathbb{R}^n \mid \|x - x_0\| \leq \delta, x_0 \in \mathbb{R}^n, \delta > 0\}$. The right derivative of a matrix function $F(x)$ is defined by $\dot{F}(x) \triangleq \lim_{h\to 0^+} \frac{F(x+h)-F(x)}{h}$. For the sake of simplicity, we denote $\mathscr{L}(A, B, P, R, \alpha) \triangleq \begin{bmatrix} A^\top P + PA + \alpha P & * \\ B^\top P & -\alpha R \end{bmatrix}$.

II. Switched Systems and Output Reachable Set

In this paper, we consider a continuous-time switched linear system in the form of

$$\Sigma:\ \dot{x}(t) = A_{\sigma(t)}x(t) + B_{\sigma(t)}u(t) \tag{1}$$
$$y(t) = C_{\sigma(t)}x(t) \tag{2}$$

where $x(t) \in \mathbb{R}^{n_x}$ are the state of the system, and the initial condition x_0 belongs to a bounded ellipsoid:

$$x_0 \in \mathcal{X}_0 \triangleq \mathcal{E}(R_0) \tag{3}$$

and $u(t) \in \mathbb{R}^{n_u}$ is the input vector which is assumed to satisfy the following ellipsoidal constraint:

$$u(t) \in \mathcal{U} \triangleq \mathcal{E}(R_u),\ \forall t \in \mathbb{R}_{\geq 0} \tag{4}$$

and $y(t) \in \mathbb{R}^{n_y}$ is the output. Define index set $\mathcal{M} \triangleq \{1, 2, \ldots, N\}$, where N is the number of modes and, $\sigma : \mathbb{R}_{\geq 0} \to \mathcal{M}$ denotes the switching function, which is assumed to be a piecewise constant function continuous from right and only non-Zeno swtichings (i.e., the switch at most a finite number of times in any finite time interval) are considered in this paper. The switching instants are expressed by a sequence $\mathcal{S} \triangleq \{t_k\}_{k\in\mathbb{N}}$, where t_0 is the initial time and t_k is the kth switching instant. Then, we define $\mathcal{I}_i \triangleq \{t \in \mathbb{R}_{\geq 0} \mid \sigma(t) = i, i \in \mathcal{M}\}$ to denote the activation time interval for ith mode. Obviously, we can see that $\bigcup_{i\in\mathcal{M}} \mathcal{I}_i = \mathbb{R}_{\geq 0}$ and $\mathcal{I}_i \cap \mathcal{I}_j = \emptyset$, for $i \neq j$, $\forall i, j \in \mathcal{M}$.

The output reachable set of system (1)–(2) is defined as

$$\mathcal{R}_y \triangleq \{y(t) \in \mathbb{R}^{n_y} \mid x(t), y(t), x_0, u(t) \text{ satisfy } (1), (2), (3), (4), t \in \mathbb{R}_{\geq 0}\} \tag{5}$$

The following lemma introduces the main idea to determine the over-approximate set $\tilde{\mathcal{R}}_y$ for switched system (1)–(2).

Lemma 1: Consider system (1)–(2) under initial state condition (3) and input condition (4). If there exist a family of Lyapunov functions $V_i : \mathbb{R}^{n_x} \to \mathbb{R}_{\geq 0}$, $i \in \mathcal{M}$, satisfying $V_i(0) = 0$ and $V_i(x) > 0$, $\forall x \neq 0$, $\forall i \in \mathcal{M}$, matrices $R_{i,y} \in \mathbb{S}_{+}^{n\times n}$, $i \in \mathcal{M}$, and scalars $\alpha > 0$, $0 < \beta \leq 1$ such that

$$F_i(t) \leq 0,\ \forall t \in \mathcal{I}_i, \forall i \in \mathcal{M} \tag{6}$$
$$G_{i,j}(t_k) \leq 0,\ \forall t_k \in \mathcal{S}, i \neq j, \forall i, j \in \mathcal{M} \tag{7}$$
$$V_i(x_0) \leq x_0^\top R_0 x_0,\ \forall i \in \mathcal{M} \tag{8}$$
$$x^\top(t)C_i^\top R_{i,y}C_i x(t) \leq V_i(x(t)),\ \forall t \in \mathcal{I}_i, \forall i \in \mathcal{M} \tag{9}$$

where $F_i(t) = \dot{V}_i(x(t)) + \alpha V_i(x(t)) - \alpha u^\top(t)R_u u(t)$ and $G_{i,j}(t_k) = V_i(x(t_k^+)) - \beta V_j(x(t_k^-)) + \beta - 1$. Then, the output reachable set $\mathcal{R}_y$ satisfies $\mathcal{R}_y \subseteq \tilde{\mathcal{R}}_y \triangleq \bigcup_{i\in\mathcal{M}} \mathcal{E}(R_{i,y})$.

Proof: See the Appendix. ■

Remark 1: Conditions (6) and (7) actually characterize an invariant set $\Omega = \bigcup_{i\in\mathcal{M}} \Omega_i$, where $\Omega_i = \{x(t) \in \mathbb{R}^{n_x} \mid V_i(x(t)) \leq 1\}$, $i \in \mathcal{M}$. By (6), it leads to $\dot{V}_i(x(t)) < 0$, $\forall x(t) \in \bar{\Omega}_i \triangleq \{x(t) \in \mathbb{R}^{n_x} \mid V_i(x(t)) > 1\}$, this guarantees that once the state $x(t)$ enters Ω_i, it remains in it during the activation time of the ith subsystem. However, (6) is not enough to ensure $x(t)$ staying in Ω forever, in presence of abrupt changes from $V_i(x(t_k^+))$ to $V_j(x(t_k^-))$, where $i \neq j$ at switching instant $t_k \in \mathcal{S}$. Thus, (7) is necessary to define the invariant Ω. It ensures that $V_i(x(t_k^+)) \leq 1$ when $V_i(x(t_k^-)) \leq 1$, that means the switching actions will not cause $x(t)$ escaping from Ω. In addition, (8) implies that the initial state $x_0 \in \mathcal{X}_0 \subseteq \bigcap_{i\in\mathcal{M}} \Omega_i$, and (9) estimates the output reachable set based on the invariant set Ω.

III. Output Reachable Set Estimation

Although Lemma 1 provides a general framework to handle the output reachable set estimation problem, it is impractical for actual use, since it does not provide any available computational techniques for the construction of Lyapunov functions $V_i(x(t))$, $i \in \mathcal{M}$. Moreover, the proposed condition (7) requires us to check the values of Lyapunov functions at all the switching instant $t_k \in \mathcal{S}$. However, the switching instant sequence $\mathcal{S}$ usually cannot be specified in advance, and it is impossible to check Lemma 1 for all the switching instants t_k in the case of $k \to \infty$. In the following, numerically tractable methods are presented to solve the output reachable set estimation problem in the framework of Lemma 1.

A. Common Lyapunov Function

One natural idea to analyze switched system (1)–(2) is to use common quadratic Lyapunov function $V_i(x(t)) = V(x(t)) = x^\top(t)Px(t)$, $i \in \mathcal{M}$, to avoid checking (7) for every $t_k \in \mathcal{S}$.

Theorem 1: Consider system (1)–(2) under initial state condition (3) and input condition (4). If there exist matrices $P \in \mathbb{S}_+^{n_x \times n_x}$, $R_{i,y} \in \mathbb{S}_+^{n\times n}$, $i \in \mathcal{M}$, and a scalar $\alpha > 0$ such that

$$\mathscr{L}(A_i, B_i, P, R_u, \alpha) \prec 0,\ \forall i \in \mathcal{M} \tag{10}$$

$$C_i^\top R_{i,y} C_i \prec P \prec R_0,\ \forall i \in \mathcal{M} \tag{11}$$

then, the output reachable set $\mathcal{R}_y \subseteq \tilde{\mathcal{R}}_y \triangleq \bigcup_{i\in\mathcal{M}} \mathcal{E}(R_{i,y})$.

Proof: Construct a Lyapunov function in the form of $V(x(t)) = x^\top(t)Px(t)$, $P \in \mathbb{S}_+^{n\times n}$. Let us consider $F(t) = \dot{V}(x(t)) + \alpha V(x(t)) - \alpha u^\top(t) R_u u(t)$, and along with the trajectory of system (1)–(2), we have $F(t) = \chi^\top(t)\mathscr{L}(A_i, B_i, P, R_u, \alpha)\chi(t)$, where $\chi^\top(t) = [x^\top(t)\ \ u^\top(t)]$, and from (10), it yields $F(t) < 0$, $\forall t \in \mathbb{R}_{\geq 0}$, so that (6) holds.

Then, since the common Lyapunov function is chosen, (7) automatically holds with $\beta = 1$. By (11), $P \prec R_0$ ensures $V(x_0) < x_0^\top R_0 x_0$, and $C_i^\top R_{i,y} C_i \prec P$, $i \in \mathcal{M}$, guarantees $x^\top(t) C_i^\top R_{i,y} C_i x(t) < V(x(t))$, $\forall t \in \mathbb{R}_{\geq 0}$, $\forall i \in \mathcal{M}$, that is (8) and (9) hold. Thus, by Lemma 1, we have the output reachable set $\mathcal{R}_y \subseteq \tilde{\mathcal{R}}_y \triangleq \bigcup_{i\in\mathcal{M}} \mathcal{E}(R_{i,y})$. ■

Remark 2: The set $\tilde{\mathcal{R}}_y$ is usually expected to be as small as possible to achieve a precise estimate of reachable set $\mathcal{R}_y$. Based on Theorem 1, one may add an additional constraint that

$$R_{i,y} \succeq \epsilon I,\ \epsilon > 0,\ \forall i \in \mathcal{M} \tag{12}$$

which implies that $\epsilon y^\top(t)y(t) \leq y^\top(t) R_{i,y} y(t) \leq 1$, namely $y(t) \in \bigcup_{i\in\mathcal{M}} \mathcal{E}(R_{i,y}) \subseteq \mathcal{B}(0, 1/\sqrt{\epsilon})$, $\forall t \in \mathbb{R}_{\geq 0}$, so we have to maximize ϵ to obtain the smallest ball $\mathcal{B}(0, 1/\sqrt{\epsilon})$ by

$$\max\ \epsilon\ \text{ s.t. } (10),\ (11) \text{ and } (12) \tag{13}$$

Moreover, due to the existence of the tuning parameter α, the result in Theorem 1 and corresponding optimization problem (13) are not standard LMI problems, they are bilinear matrix inequality (BMI) problems and known to be NP-hard. Fortunately, several algorithms are available to solve BMI problems such as the iterative linear matrix inequality (ILMI) approach in [25], [26], or using numerical optimization algorithms, such as `fminsearch` [13] or genetic algorithm (GA) [18] in the optimization toolbox of Matlab.

B. Multiple Lyapunov Functions

Switching actions are able to significantly affect the evolution of switched systems, for example the instability arises as a result of a rapid switching between stable subsystems. Similarly, the switching rate has a great impact on the reachable set as well. Thus, given a switching rate, how to estimate the set $\mathcal{R}_y$ is one of the basic problems for reachable set estimation. In this work, the concept of minimum dwell time is given to constrain the switching rate.

Definition 1: [27] Given a switching signal function $\sigma(t)$ with a generated switching sequence $\mathcal{S}$, $\tau_{\min} = \inf_{k\in\mathbb{N}}\{t_{k+1} - t_k\}$ is called the dwell time of $\sigma(t)$, and $\mathcal{D}_{\tau_{\min}} \triangleq \{\sigma(t) \mid \sigma : \mathbb{R}_{\geq 0} \to \mathcal{M}, t_{k+1} - t_k \geq \tau_{\min}, \forall k \in \mathbb{N}\}$ denotes the set of all switching policies with dwell time greater than $\tau_{\min}$.

We consider a class of time-scheduled multiple Lyapunov functions inspired by [28]–[31] as follows:

$$V_i(x(t)) = x^\top(t)P_i(t)x(t),\ t \in \mathbb{R}_{\geq 0},\ i \in \mathcal{M} \tag{14}$$

where $P_i(t) \in \mathbb{S}_+^{n\times n}$, $i \in \mathcal{M}$ have the following structure:

Consider the interval $[t_k, t_k + \tau_{\min})$, we partition it into L segments described as $\mathcal{L}_{k,q} \triangleq [t_k + \theta_q, t_k + \theta_{q+1})$, $q = 0, 1, \ldots, L-1$ of equal lengths $h = \tau_{\min}/L$, and then $\theta_0 = 0$ and $\theta_q = qh = q\tau_{\min}/L$. We consider a class of continuous matrix function $P_i(t)$, $t \in [t_k, t_k + \tau_{\min})$ chosen to be linear within each segment $\mathcal{L}_{k,q}$, $q = 0, 1, \ldots, L-1$. Explicitly, we can see that $\bigcup_{n=0}^{L-1} \mathcal{L}_{k,n} = [t_k, t_k + \tau_{\min})$ and $\mathcal{L}_{k,n} \cap \mathcal{L}_{k,m} = \emptyset$, $n \neq m$. Letting $P_{i,q} = P_i(t_k + \theta_q)$, then since the matrix function $P_i(t)$ is piecewise linear in $[t_k, t_k + \tau_{\min})$, it can be expressed in terms of the values at dividing points using a linear interpolation formula, that is, for $0 \leq \mu \leq 1$, $q = 0, 1, \ldots, L-1$,

$$P_i(t) = P_i(\mu) = (1-\mu)P_{i,q} + \mu P_{i,q+1},\ t \in \mathcal{L}_{k,q},\ i \in \mathcal{M} \tag{15}$$

where $\mu = L(t - t_k - \theta_q)/\tau_{\min}$.

As a result, the continuous matrix function $P_i(t) \in \mathbb{S}_+^{n\times n}$, $i \in \mathcal{M}$ can be completely determined by $P_{i,q} \in \mathbb{S}_+^{n\times n}$, $q = 0, 1, \ldots, L$, $i \in \mathcal{M}$, in interval $[t_k, t_k + \tau_{\min})$. Then, due to $[t_k, t_k + \tau_{\min}) \subseteq [t_k, t_{k+1})$, for the remaining time in $[t_k, t_{k+1})$ denoted by $\mathcal{L}_{k,L} \triangleq [t_{k,\min}, t_{k+1})$, $P_i(t)$, $i \in \mathcal{M}$ is set to be

$$P_i(t) = P_{i,L},\ t \in \mathcal{L}_{k,L},\ i \in \mathcal{M} \tag{16}$$

In summary, $P_i(t)$, $i \in \mathcal{M}$ is defined as

$$P_i(t) = \begin{cases} P_i(\mu), & t \in \mathcal{L}_{k,q},\ q = 0, 1, \ldots, L-1 \\ P_{i,L}, & t \in \mathcal{L}_{k,L} \end{cases} \tag{17}$$

where μ is defined in (15).

Theorem 2: Given a dwell time $\tau_{\min} > 0$ and consider switched system (1)–(2) with $\sigma(t) \in \mathcal{D}_{\tau_{\min}}$ under initial state condition (3) and input condition (4). If there exist matrices $P_{i,q} \in \mathbb{S}_+^{n_x \times n_x}$, $q = 0, 1, \ldots, L$, $i \in \mathcal{M}$, $R_{i,y} \in \mathbb{S}_+^{n\times n}$, $i \in \mathcal{M}$, and a scalar $\alpha > 0$ such that for $\forall i, j \in \mathcal{M}$

$$\mathscr{L}(A_i, B_i, P_{i,q}, R_u, \alpha) + \Psi_{i,q} \prec 0,\ q = 0, \ldots, L-1 \tag{18}$$

$$\mathscr{L}(A_i, B_i, P_{i,q+1}, R_u, \alpha) + \Psi_{i,q} \prec 0,\ q = 0, \ldots, L-1 \tag{19}$$

$$\mathscr{L}(A_i, B_i, P_{i,L}, R_u, \alpha) \prec 0 \tag{20}$$

$$P_{i,0} - P_{j,L} \prec 0,\ i \neq j \tag{21}$$

$$P_{i,0} - R_0 \prec 0 \tag{22}$$

$$C_i^\top R_{i,y} C_i - P_{i,q} \prec 0,\ q = 0, \ldots, L \tag{23}$$

where $\Psi_{i,q} = \operatorname{diag}\{L(P_{i,q+1} - P_{i,q})/\tau_{\min}, 0\}$. Then, the output reachable set $\mathcal{R}_y \subseteq \tilde{\mathcal{R}}_y \triangleq \bigcup_{i\in\mathcal{M}} \mathcal{E}(R_{i,y})$.

Proof: Construct a Lyapunov function as $V(t) = \sum_{i\in\mathcal{M}} \xi_i(t) x^\top(t) P_i(t) x(t)$, where $P_i(t)$, $i \in \mathcal{M}$, is defined by (17) and $\xi_i : \mathbb{R}_{\geq 0} \to \{0, 1\}$ and $\sum_{i\in\mathcal{M}} \xi_i(t) = 1$ is the indicator function representing the active modes at time t.

TABLE I
COMPUTATIONAL COMPLEXITIES OF THEOREM 2 WITH A FIXED α

Number of Decision Variables	LMI Constraints Size
$nN(L+1)(n+1)/2$	$n(N^2+2N+3L)$

First, let us consider $F_i(t) = \dot{V}(t) + \alpha V(t) - \alpha u^\top(t) R_u u(t)$, which can be rewritten to

$$F_i(t) = \chi^\top(t)(\mathscr{L}(A_i, B_i, P_i(t), R_u, \alpha) + \Psi_i(t))\chi(t) \quad (24)$$

where $\chi^\top(t) = [x^\top(t) \quad u^\top(t)]$ and $\Psi_i(t) = \mathrm{diag}\{\dot{P}_i(t), 0\}$.

Suppose $\sigma(t) = i$, $t \in \mathcal{L}_{k,q}$, $q = 0, \dots, L-1$, one has

$$\mathscr{L}(A_i, B_i, P_i(t), R_u, \alpha) = (1-\mu)\Xi_{i,1} + \mu\Xi_{i,2} \quad (25)$$

where $\Xi_{i,1} = \mathscr{L}(A_i, B_i, P_{i,q}, R_u, \alpha)$ and $\Xi_{i,2} = \mathscr{L}(A_i, B_i, P_{i,q+1}, R_u, \alpha)$. Furthermore, we can see that $\dot{P}_i(t) = (P_{i,q+1} - P_{i,q})\dot{\mu}$, $t \in \mathcal{L}_{k,q}$, $q = 0, \dots, L-1$, and because of $\mu = L(t - t_k - \theta_q)/\tau_{\min}$, it implies that $\dot{\mu} = L/\tau_{\min}$, leading to $\dot{P}_i(t) = \Psi_{i,q}$, $t \in \mathcal{L}_{k,q}$, $q = 0, \dots, L-1$. Thus, by (18), (19), it leads to

$$F_i(t) < 0,\ \forall t \in \bigcup\nolimits_{n=0}^{L-1} \mathcal{L}_{k,n} = [t_k, t_k + \tau_{\min}) \quad (26)$$

Then, we consider $t \in \mathcal{L}_{k,L}$. Since $P_i(t) = P_{i,L}$, $t \in \mathcal{L}_{k,L}$, we have $P_i(t) = 0$, $\forall t \in \mathcal{L}_{k,L}$, thus (20) guarantees that

$$F_i(t) < 0,\ \forall t \in \mathcal{L}_{k,L} \quad (27)$$

Thus, from (26) and (27), we can conclude that $F_i(t) < 0$, $\forall t \in \mathcal{I}_i$, $\forall i \in \mathcal{M}$, which means (6) in Lemma 1 holds. Next, (21) ensures (7) holds with $\beta = 1$ and (22) guarantees (8) holds Finally, we consider

$$\begin{aligned} &C_i^\top R_{i,y} C_i - P_i(t) \\ =&(1-\mu)(C_i^\top R_{i,y} C_i - P_{i,q}) + \mu(C_i^\top R_{i,y} C_i - P_{i,q+1}) \end{aligned}$$

and (23) ensures that $C_i^\top R_{i,y} C_i - P_i(t) < 0$, $\forall t \in \mathbb{R}_{\geq 0}$, $\forall i \in \mathcal{M}$, which implies (9) holds. Therefore, we have the output reachable set $\mathcal{R}_y \subseteq \tilde{\mathcal{R}}_y \triangleq \bigcup_{i \in \mathcal{M}} \mathcal{E}(R_{i,y})$ by Lemma 1. ■

Remark 3: Some remarks on parameter L are given.

(1) Parameter L implies the number of segments consisting of the dwell time interval $[t_k, t_k + \tau_{\min})$. A larger L yields a finer division of $[t_k, t_k + \tau_{\min})$, and a less conservative result can be consequently obtained, which will be demonstrated by a numerical example later. However, the computational cost increases as L grows, since a larger L inevitably introduces more decision variables and LMI constraints, see TABLE I for the computational complexity analysis for Theorem 2 for an n-dimensional switched system consisting of N modes.

(2) Similar as the methods adopted in [20], a piecewise matrix function $P_i(\mu)$ in (15) with a sufficiently large L is able to approximate a generic continuously differentiable $P_i(t)$ with adequate accuracy over the finite-time interval $[t_k, t_k + \tau_{\min})$. In other words, if $L \to \infty$, conditions (18)–(23) in Theorem 2 can be expressed as follows with $i, j \in \mathcal{M}$ and $t \in [0, \tau_{\min})$

$$P_i(t) \succ 0 \quad (28)$$

$$\mathscr{L}(A_i, B_i, P_i(t), R_u, \alpha) + \Psi_i(t) \prec 0 \quad (29)$$

$$\mathscr{L}(A_i, B_i, P_i(\tau_{\min}), R_u, \alpha) \prec 0 \quad (30)$$

$$P_i(0) - P_j(\tau_{\min}) \prec 0,\ i \neq j \quad (31)$$

$$P_i(0) - R_0 \prec 0 \quad (32)$$

$$C_i^\top R_{i,y} C_i - P_i(t) \prec 0 \quad (33)$$

where $\Psi_i(t) = \mathrm{diag}\{\dot{P}_i(t), 0\}$. It should be noted that the above differential linear matrix inequality (DLMI) (28)–(33) can achieve the result with least conservativeness in our framework, but it is not numerically tractable due to the presence of continuous matrix functions $P_i(t)$.

(3) In another extreme case with $L = 0$, $P_{i,q}$, shrinks to P_i, moreover, due to (21), we have to choose $P_i = P_j$, $i \neq j$. Thus, Theorem 2 is reduced to Theorem 1, namely the common Lyapunov function result.

Given an L, the smallest ball $\mathcal{B}(0, 1/\sqrt{\epsilon})$ containing the trajectories of output $y(t)$ in the framework of our approach can be obtained. Based on Theorem 2, an optimization problem can be formulated by adding (12) with (18)–(23) as follows:

$$\max\ \epsilon \ \text{ s.t. (12) and (18)} - \text{(23)} \quad (34)$$

C. Example

Consider a switched system with two subsystems as

$$\left[\begin{array}{c} A_1 \\ \hdashline B_1^\top \\ \hdashline C_1 \end{array}\right] = \left[\begin{array}{cc} -2 & 1 \\ 0 & -0.9 \\ \hdashline 3 & 1 \\ \hdashline 1 & 0 \\ 0 & 1 \end{array}\right], \quad \left[\begin{array}{c} A_2 \\ \hdashline B_2^\top \\ \hdashline C_2 \end{array}\right] = \left[\begin{array}{cc} -1 & 0 \\ -1 & -1 \\ \hdashline 2 & 3 \\ \hdashline 1 & 0 \\ 0 & 1 \end{array}\right]$$

The initial state is assumed to satisfy $x_0 \in \{x_0 \in \mathbb{R}^2 \mid \|x_0\| \leq 1\}$ and the input is assumed to satisfy $u(t) \in \{u(t) \in \mathbb{R} \mid -1 \leq u(t) \leq 1,\ \forall t \in \mathbb{R}_{\geq 0}\}$, which implies that $R_0 = \mathrm{diag}\{1, 1\}$ and $R_u = 1$.

First, we use Theorem 1 to estimate the reachable set $\tilde{\mathcal{R}}_y$ contained in the ball $\mathcal{B}(0, \delta)$ with the minimal δ, where $\delta = 1/\sqrt{\epsilon}$. The minimal δ is 2.9033 obtained by solving optimization (13) with the aid of `fminsearch`. It should be noted that this result is applicable for the arbitrary switching, since the common Lyapunov function approach is employed.

Next, if the dwell-time constraint is further considered in the switching signal, we can apply Theorem 2. Suppose dwell time $\tau_{\min} = 1$, we solve optimization problem (34) to obtain the minimal δ with $L = 1, 2, \dots, 10$, which are depicted in Fig. 1. The following two points can be observed in Fig. 1, which are consistent with Remark 3.

1) The value of δ monotonically decreases as L increases. This means that a less conservative result, namely a smaller δ, can be obtained, if a greater L is chosen.

2) The $L = 0$ is equivalent to the result of common Lyapunov function approach, but it is more restrictive than the result obtained by the multiple Lyapunov function approach with $L \geq 1$.

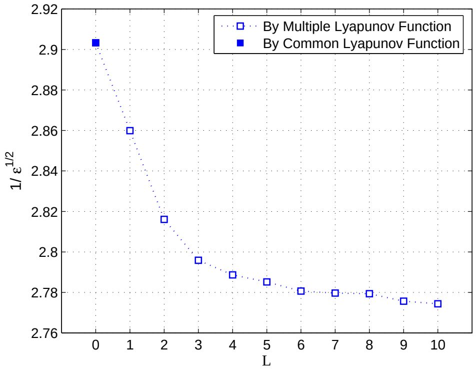


Fig. 1. Minimized $\delta = 1/\sqrt{\epsilon}$ by the common Lyapunov function approach (Theorem 1) and the multiple Lyapunov function approach (Theorem 2) with respect to $L = 0, 1, 2, \dots, 10$.

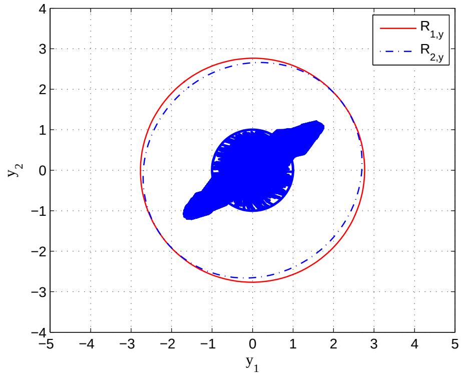


Fig. 2. 1000 randomly generated state trajectories are bounded in the estimated output reachable set $\tilde{\mathcal{R}}_y = \mathcal{R}_{1,y} \cup \mathcal{R}_{2,y}$.

Finally, the bounding ellipsoids $\mathcal{R}_{1,y}$ and $\mathcal{R}_{2,y}$ obtained with $L = 10$ are shown in Fig. 2. The switching signal has $t_{k+1} - t_k = 1 + \mathrm{rand}$, $k \in \mathbb{N}$, where rand is a random number within $[0, 1]$, thus the switching signal $\sigma(t) \in D_{\tau_{\min}}$ with $\tau_{\min} = 1$. With an input $u(t) = \sin(t)$, 1000 state trajectories generated from 1000 random initial states from a unit circle are illustrated in Fig. 2. As Fig. 2 shows, all the state trajectories are bounded in the estimated reachable set $\tilde{\mathcal{R}}_y = \mathcal{R}_{1,y} \cup \mathcal{R}_{2,y}$, showing the effectiveness of our approach.

IV. Safety Verification for Uncertain Switched System

For the sake of being concise, we focus on the application of Theorem 2 in the rest of this paper, since Theorem 1 is just a special case of Theorem 2 with parameter $L = 0$, see point (3) in Remark 3.

A. Approximate Bisimulation

For a continuous-time switched linear system Σ described by (1)–(2), an approximately bisimilar continuous-time switched linear system $\tilde{\Sigma}$ is considered in the following form

$$\tilde{\Sigma}: \ \dot{\tilde{x}}(t) = \tilde{A}_{\sigma(t)}\tilde{x}(t) + \tilde{B}_{\sigma(t)}u(t) \tag{35}$$

$$\tilde{y}(t) = \tilde{C}_{\sigma(t)}\tilde{x}(t) \tag{36}$$

where $\tilde{x}(t) \in \mathbb{R}^{\tilde{n}_x}$ is the state of the bisimilar system, the initial state $\tilde{x}_0$ is assumed to be in

$$\tilde{x}_0 \in \tilde{\mathcal{X}}_0 \triangleq \mathcal{E}(\tilde{R}_0) \tag{37}$$

and $\tilde{y}(t) \in \mathbb{R}^{n_y}$ is the output of the bisimilar system. In the rest of the work, the input $u(t)$ and switching signal $\sigma(t)$ of $\tilde{\Sigma}$ is considered to be same as those for system Σ.

Definition 2: [22] A relation $\mathscr{R}_\delta \subseteq \mathbb{R}^{n_x} \times \mathbb{R}^{\tilde{n}_x}$ is called a δ-approximate bisimulation relation between systems Σ and $\tilde{\Sigma}$, of precision δ, if for all $(x(t), \tilde{x}(t)) \in \mathscr{R}_\delta$

1) $\|y(t) - \tilde{y}(t)\| \leq \delta$, $\forall t \in \mathbb{R}_{\geq 0}$,
2) $\forall u(t) \in \mathcal{U}$, $\forall x(t)$ satisfies Σ, $\exists \tilde{x}(t)$ satisfies $\tilde{\Sigma}$ such that $(x(t), \tilde{x}(t)) \in \mathscr{R}_\delta$, $\forall t \in \mathbb{R}_{\geq 0}$,
3) $\forall u(t) \in \mathcal{U}$, $\forall \tilde{x}(t)$ satisfies $\tilde{\Sigma}$, $\exists x(t)$ satisfies Σ such that $(x(t), \tilde{x}(t)) \in \mathscr{R}_\delta$, $\forall t \in \mathbb{R}_{\geq 0}$.

and we say systems Σ and $\tilde{\Sigma}$ are approximately bisimilar with precision δ, denoted by $\Sigma \sim_\delta \tilde{\Sigma}$.

Define the following notations $\hat{x}(t) = [x^\top(t) \ \tilde{x}^\top(t)]^\top$, $\hat{y}(t) = y(t) - \tilde{y}(t)$ and

$$\left[\begin{array}{c:c:c} \hat{A}_i & \hat{B}_i & \hat{C}_i^\top \end{array}\right] = \left[\begin{array}{cc:c:c} A_i & 0 & B_i & C_i^\top \\ 0 & \tilde{A}_i & \tilde{B}_i & -\tilde{C}_i^\top \end{array}\right]$$

and let $0 \leq \gamma \leq 1$, we define $\hat{R}_0(\gamma) = \mathrm{diag}\{\gamma R_0, (1-\gamma)\tilde{R}_0\}$.

Since Σ and $\tilde{\Sigma}$ share same switching signal $\sigma(t)$ and input $u(t)$, an augmented system $\hat{\Sigma}$ can be derived from Σ and $\tilde{\Sigma}$ as below

$$\hat{\Sigma}: \ \dot{\hat{x}}(t) = \hat{A}_{\sigma(t)}\hat{x}(t) + \hat{B}_{\sigma(t)}u(t) \tag{38}$$

$$\hat{y}(t) = \hat{C}_{\sigma(t)}\hat{x}(t) \tag{39}$$

with initial state $\hat{x}_0 \in \hat{\mathcal{X}}_0 \triangleq \mathcal{E}(\hat{R}_0(\gamma))$ and input $u(t) \in \mathcal{U} \triangleq \mathcal{E}(R_u)$.

Because $\|y(t) - \tilde{y}(t)\| \leq \delta$, $\forall t \in \mathbb{R}_{\geq 0}$ holds if and only if $\hat{y}(t) \in \mathcal{B}(0, \delta)$, $\forall t \in \mathbb{R}_{\geq 0}$, the problem of computing the distance δ between Σ and $\tilde{\Sigma}$ can be converted to the problem of output reachable set estimation for augmented system $\hat{\Sigma}$.

Theorem 3: Given a dwell time $\tau_{\min} > 0$ and consider switched systems Σ by (1)–(2) and $\tilde{\Sigma}$ by (35)–(36) with $\sigma(t) \in \mathcal{D}_{\tau_{\min}}$ under initial state condition (3), (37) and input condition (4). If there exist a set of matrices $P_{i,q} \in \mathbb{S}_+^{(n_x+\tilde{n}_x)\times(n_x+\tilde{n}_x)}$, $q = 0, 1, \dots, L$, $i \in \mathcal{M}$ and scalars $\alpha > 0$, $0 \leq \gamma \leq 1$, $\epsilon \geq 0$ such that for $\forall i, j \in \mathcal{M}$

$$\mathscr{L}(\hat{A}_i, \hat{B}_i, P_{i,q}, R_u, \alpha) + \Psi_{i,q} \prec 0, \ q = 0, \dots, L-1 \tag{40}$$

$$\mathscr{L}(\hat{A}_i, \hat{B}_i, P_{i,q+1}, R_u, \alpha) + \Psi_{i,q} \prec 0, \ q = 0, \dots, L-1 \tag{41}$$

$$\mathscr{L}(\hat{A}_i, \hat{B}_i, P_{i,L}, R_u, \alpha) \prec 0 \tag{42}$$

$$P_{i,0} - P_{j,L} \prec 0, \ i \neq j \tag{43}$$

$$P_{i,0} - \hat{R}_0(\gamma) \prec 0 \tag{44}$$

$$\epsilon \hat{C}_i^\top \hat{C}_i - P_{i,q} \prec 0, \ q = 0, \dots, L \tag{45}$$

where $\Psi_{i,q} = \mathrm{diag}\{L(P_{i,q+1} - P_{i,q})/\tau_{\min}, 0\}$. Then, we have an approximation bisimulation relation $\mathscr{R}_\delta$, $\delta = 1/\sqrt{\epsilon}$ such that $\Sigma \sim_\delta \tilde{\Sigma}$.

Proof: Since the initial states $x_0 \in \mathcal{E}(R_0)$ and $\tilde{x}_0 \in \mathcal{E}(\tilde{R}_0)$, the initial state $\hat{x}_0$ satisfies

$$\hat{x}_0^\top \hat{R}_0(\gamma)\hat{x}_0 = \gamma x_0^\top R_0 x_0 + (1-\gamma)\tilde{x}_0^\top \tilde{R}_0 \tilde{x}_0 \leq 1, \ 0 \leq \gamma \leq 1 \tag{46}$$

Thus, it means that $\hat{x}_0 \in \mathcal{E}(\hat{R}_0(\gamma))$, $0 \le \gamma \le 1$.

From Theorem 2, it implies that the output reachable set of $\hat{\Sigma}$ can be estimated by $\bigcup_{i\in\mathcal{M}} \mathcal{E}(\epsilon I) = \mathcal{B}(0,\delta)$, where $\delta = 1/\sqrt{\epsilon}$, so the output $\hat{y}(t) \in \mathcal{B}(0,\delta)$, $\forall t \in \mathbb{R}_{\ge 0}$. Furthermore, due to $\hat{y}(t) = y(t) - \tilde{y}(t)$, we have $\|y(t) - \tilde{y}(t)\| \le \delta$, $\forall t \in \mathbb{R}_{\ge 0}$, along with the trajectories $x(t)$, $\tilde{x}(t)$ generated by Σ and $\tilde{\Sigma}$. The approximation bisimulation relation $\mathscr{R}_\delta$ such that $\Sigma \sim_\delta \tilde{\Sigma}$ can be established. ■

The choice of a larger L in Theorem 3 will lead to a less conservative analysis result, the result with the least conservativeness can be deduced by letting $L \to \infty$, which is however numerically intractable. For the particular case with $L = 0$, Theorem 3 is reduced to a result by the common Lyapunov function approach, but it can be used for the arbitrary switching case.

B. Safety Verification

We consider the system matrices of switched system Σ are uncertain and satisfy that $[A_i \;\; B_i \;\; C_i^\top] \in \mathfrak{R}_i$, where

$$\mathfrak{R}_i \triangleq \mathbf{co}\left\{[A_i^{(1)}\; B_i^{(1)}\; (C_i^{(1)})^\top], \ldots, [A_i^{(S)}\; B_i^{(S)}\; (C_i^{(S)})^\top]\right\} \tag{47}$$

where $\mathbf{co}\{\cdot\}$ is the convex-hull operator.

Definition 3: Consider system Σ described by (1)–(2) and (47) with $C_i^{(s)} = I$, $\forall s = 1, \ldots, S, \forall i \in \mathcal{M}$. System Σ is said to be safe with respect to the unsafe region Ω_u, if $\mathcal{R}_y \cap \Omega_u = \emptyset$.

Let $\tilde{\Sigma}$ be an approximately bisimilar system such that $\Sigma \sim_\delta \tilde{\Sigma}$. Denote $\mathcal{R}_y$, $\mathcal{R}_{\tilde{y}}$ the output reachable sets of Σ and $\tilde{\Sigma}$ respectively, then it can be seen that $\mathcal{R}_y \subseteq \mathcal{N}(\mathcal{R}_{\tilde{y}}, \delta)$, where $\mathcal{N}(\cdot, \delta)$ denotes the δ-neighborhood of a set. Consequently, to prove that Σ is safe, it is sufficient to verify that $\mathcal{R}_{\tilde{y}} \cap \mathcal{N}(\Omega_u, \delta) = \emptyset$.

Proposition 1: If $\Sigma \sim_\delta \tilde{\Sigma}$, then $\mathcal{R}_{\tilde{y}} \cap \mathcal{N}(\Omega_u, \delta) = \emptyset \Rightarrow \mathcal{R}_y \cap \Omega_u = \emptyset$. Namely, $\tilde{\Sigma}$ is safe with respect to $\mathcal{N}(\Omega_u, \delta) \Rightarrow \Sigma$ is safe with respect to Ω_u.

In the following, a theorem is presented to compute the system matrices for a bisimilar system for uncertain switched system Σ.

Theorem 4: Given a dwell time $\tau_{\min} > 0$ and consider uncertain switched systems Σ by (1)–(2), (47) and $\tilde{\Sigma}$ by (35)–(36) with $\sigma(t) \in \mathcal{D}_{\tau_{\min}}$ under initial state condition (3), (37) and input condition (4). If there exist a set of matrices $M_i \in \mathbb{R}^{n_x \times n_x}$, $N_i \in \mathbb{R}^{n_x \times n_u}$, $X_i \in \mathbb{R}^{n_x \times n_x}$, $Y_i \in \mathbb{R}^{n_x \times n_x}$, $Z_i \in \mathbb{R}^{n_x \times n_x}$, $S_i \in \mathbb{R}^{n_y \times n_x}$, $P_{i,q} \in \mathbb{S}_+^{2n_x \times 2n_x}$, $q = 0, 1, \ldots, L$, $i \in \mathcal{M}$ and scalars $\alpha > 0$, $0 \le \gamma \le 1$, $\delta \ge 0$ such that for $\forall i, j \in \mathcal{M}$ and $\forall s = 1, 2, \ldots, S$,

$$\Xi_{i,q,1}^{(s)} \prec 0,\; q = 0, \ldots, L-1 \tag{48}$$

$$\Xi_{i,q,2}^{(s)} \prec 0,\; q = 0, \ldots, L-1 \tag{49}$$

$$\Xi_{i,L}^{(s)} \prec 0 \tag{50}$$

$$P_{i,0} - P_{j,L} \prec 0,\; i \neq j \tag{51}$$

$$P_{i,0} - \hat{R}_0(\gamma) \prec 0 \tag{52}$$

$$\begin{bmatrix} -P_{i,q} & * \\ W_i^{(s)} & -\delta^2 I \end{bmatrix} \prec 0,\; q = 0, \ldots, L \tag{53}$$

where $\hat{R}_0(\gamma) = \mathrm{diag}\{\gamma R_0, (1-\gamma)R_0\}$, and

$$\Xi_{i,q,1}^{(s)} = \begin{bmatrix} -\mathrm{Sym}(U_i^{(s)}) + \alpha P_{i,q} + \Psi_{i,q} & * & * \\ -(V_i^{(s)})^\top & -\alpha R_u & * \\ P_{i,q} + Q_i - (U_i^{(s)})^\top & -V_i^{(s)} & \mathrm{Sym}(Q_i) \end{bmatrix}$$

$$\Xi_{i,q,2}^{(s)} = \begin{bmatrix} -\mathrm{Sym}(U_i^{(s)}) + \alpha P_{i,q+1} + \Psi_{i,q} & * & * \\ -(V_i^{(s)})^\top & -\alpha R_u & * \\ P_{i,q+1} + Q_i - (U_i^{(s)})^\top & -V_i^{(s)} & \mathrm{Sym}(Q_i) \end{bmatrix}$$

$$\Psi_{i,q} = L(P_{i,q+1} - P_{i,q})/\tau_{\min}$$

$$\Xi_{i,L}^{(s)} = \begin{bmatrix} -\mathrm{Sym}(U_i^{(s)}) + \alpha P_{i,L} & * & * \\ -(V_i^{(s)})^\top & -\alpha R_u & * \\ P_{i,L} + Q_i - (U_i^{(s)})^\top & -V_i^{(s)} & \mathrm{Sym}(Q_i) \end{bmatrix}$$

$$U_i^{(s)} = \begin{bmatrix} X_i A_i^{(s)} & M_i \\ Z_i A_i^{(s)} & M_i \end{bmatrix},\; V_i^{(s)} = \begin{bmatrix} X_i B_i^{(s)} + N_i \\ Z_i B_i^{(s)} + N_i \end{bmatrix}$$

$$W_i^{(s)} = \begin{bmatrix} C_i^{(s)} & -S_i^\top \end{bmatrix},\; Q_i = \begin{bmatrix} X_i & Y_i \\ Z_i & Y_i \end{bmatrix}$$

Then, we can obtain an approximately bisimilar system $\tilde{\Sigma}$ in the form of (35)–(36) and an approximation bisimulation relation $\mathscr{R}_\delta$ such that $\Sigma \sim_\delta \tilde{\Sigma}$, where the corresponding system matrices are

$$\left[\begin{array}{c:c:c} \tilde{A}_i & \tilde{B}_i & \tilde{C}_i^\top \end{array}\right] = \left[\begin{array}{c:c:c} Y_i^{-1} M_i & Y_i^{-1} N_i & S_i^\top \end{array}\right] \tag{54}$$

Proof: First, $Q_i + Q_i^\top \prec 0$ implies $Y_i + Y_i^\top \prec 0$, thus Y_i is nonsingular. Then, substituting $M_i = \tilde{A}_i Y_i$, $N_i = \tilde{B}_i Y_i$ and $S_i = \tilde{C}_i$ into (48), it becomes

$$\begin{bmatrix} -\mathrm{Sym}(Q_i \hat{A}_i^{(s)}) + \alpha P_{i,q} + \Psi_{i,q} & * & * \\ -(\hat{B}_i^{(s)})^\top Q_i^\top & -\alpha R_u & * \\ P_{i,q} + Q_i - (\hat{A}_i^{(s)})^\top Q_i^\top & -Q_i \hat{B}_i^{(s)} & Q_i + Q_i^\top \end{bmatrix} \prec 0$$

By left-multiplying the third row of above inequality by $(\hat{A}_i^{(s)})^\top$ or $(\hat{B}_i^{(s)})^\top$ and adding it to the first or second row, and right-multiplying the third column by $\hat{A}_i^{(s)}$ or $\hat{B}_i^{(s)}$ and adding it to the first or second column, it yields

$$\begin{bmatrix} \mathrm{Sym}(P_{i,q} \hat{A}_i^{(s)}) + \alpha P_{i,q} + \Psi_{i,q} & * & * \\ (\hat{B}_i^{(s)})^\top P_{i,q} & -\alpha R_u & * \\ P_{i,q} + Q_i^\top - Q_i \hat{A}_i^{(s)} & -Q_i \hat{B}_i^{(s)} & Q_i + Q_i^\top \end{bmatrix} \prec 0$$

Due to (47) and simple convexity arguments, the above inequality ensures (40) holds. Through a similar proof, it can be found that (49) $\Rightarrow$ (41) and (50) $\Rightarrow$ (42). Moreover, (51) and (52) are equivalent to (43) and (44).

Finally, letting $\epsilon = 1/\delta^2$ and by Schur complement, (53) ensures that (45) holds. Therefore, the approximation bisimulation $\Sigma \sim_\delta \tilde{\Sigma}$ can be established by Theorem 3. ■

Given an L, the optimized approximately bisimilar system $\tilde{\Sigma}_{\mathrm{opt}}$ can be obtained by minimizing the precision δ by

$$\min\ \delta^2 \quad \text{s.t.}\ (48) - (53) \tag{55}$$

So far, according to Proposition 1, we can perform the safety verification for uncertain system Σ with respect to Ω_u via verifying the safety specification of the bisimilar system $\tilde{\Sigma}$ with respect to the set $\mathcal{N}(\Omega_u, \delta)$, the δ-neighborhood of Ω_u.

TABLE II
PRECISION δ WITH $L = 1, 2, 3, 4, 5$ AND COMPUTATION TIME (C.T.) WITH A FIXED α

	$L = 1$	$L = 2$	$L = 3$	$L = 4$	$L = 5$
δ	0.459	0.434	0.425	0.414	0.409
C. T.	0.573 s	0.862 s	1.221 s	4.762 s	15.263 s

C. Example

In this subsection, the safety verification for an uncertain switched affine system $\dot{x}(t) = A_i(t)x + b_i$, $i \in \{1, 2\}$, is considered. The system matrices are given as blow:

$$\left[\begin{array}{c} A_1 \\ \hdashline b_1^\top \end{array}\right] = \left[\begin{array}{cc} -2 & 1 \\ \gamma(t) & -0.9 \\ \hdashline 3 & 1 \end{array}\right], \quad \left[\begin{array}{c} A_2 \\ \hdashline b_2^\top \end{array}\right] = \left[\begin{array}{cc} -1 & \gamma(t) \\ -1 & -1 \\ \hdashline 2 & 3 \end{array}\right]$$

where $\gamma(t) \in [0, 0.1]$ is an uncertain time-varying parameter. The initial state is assumed to be $x_0 \in \{x_0 \in \mathbb{R}^2 \mid \|x_0\| \leq 0.1\}$, which implies that $R_0 = \text{diag}\{100, 100\}$. The switching signal is a periodic switching law as $t_{k+1} - t_k = 1$, $\forall k \in \mathbb{N}$.

Using Theorem 4, a switched system with exact parameters can be obtained, with a corresponding precision δ. One point needs to be clarified here is that (50) can be removed for this particular periodic switching case, since (50) exactly corresponds to the interval $[t_k + \tau_{\min}, \infty)$ which does not appear at all. Similar to the experimental results for reachable set estimation (Section III, C), the precision δ tends to a smaller value as a larger L is chosen to apply Theorem 4, see TABLE II for $L = 1, 2, 3, 4, 5$.

Then, in order to validate our approach, we first let $L = 1$ and obtain the corresponding system matrices as follows:

$$\left[\begin{array}{c} A_1 \\ \hdashline b_1^\top \\ \hdashline C_1 \end{array}\right] = \left[\begin{array}{cc} -1.520 & 0.383 \\ 0.152 & -1.115 \\ \hdashline -2.895 & -1.087 \\ \hdashline -0.969 & -0.036 \\ -0.0355 & -0.9413 \end{array}\right], \left[\begin{array}{c} A_2 \\ \hdashline b_2^\top \\ \hdashline C_2 \end{array}\right] = \left[\begin{array}{cc} -0.858 & -0.091 \\ -0.508 & -1.560 \\ \hdashline -2.043 & -3.112 \\ \hdashline -0.969 & -0.036 \\ -0.036 & -0.941 \end{array}\right]$$

With the above switched system with exact parameters, we can conduct the verification for the uncertain switched system. Given three unsafe regions $\Omega_{u,1} \triangleq \mathcal{B}([0.7\ 1.7], 0.6)$, $\Omega_{u,2} \triangleq \mathcal{B}([2\ -0.2], 0.5)$ and $\Omega_{u,3} \triangleq \mathcal{B}([3.5\ 1.5], 0.9)$, the new unsafe regions are described by their neighborhoods $\tilde{\Omega}_{u,1} \triangleq \mathcal{N}(\Omega_{u,1}, 0.459)$, $\tilde{\Omega}_{u,2} \triangleq \mathcal{N}(\Omega_{u,2}, 0.459)$ and $\tilde{\Omega}_{u,3} \triangleq \mathcal{N}(\Omega_{u,3}, 0.459)$. Thus, the verification for uncertain switched system can be done via verifying if the new system is safe with respect to the new unsafe regions. We can use SpaceEx [32] to perform the verification for the certain system.

The verification result is illustrated in Fig. 3. However, the safety of the original system cannot be guaranteed since the computed reach set intersects with $\tilde{\Omega}_{u,3}$. Then, we let $L = 5$ which produces a smaller precision δ, and the system matrices are

$$\left[\begin{array}{c} A_1 \\ \hdashline b_1^\top \\ \hdashline C_1 \end{array}\right] = \left[\begin{array}{cc} -1.611 & 0.427 \\ 0.170 & -1.106 \\ \hdashline -2.981 & -1.030 \\ \hdashline -0.970 & -0.032 \\ -0.032 & -0.948 \end{array}\right], \left[\begin{array}{c} A_2 \\ \hdashline b_2^\top \\ \hdashline C_2 \end{array}\right] = \left[\begin{array}{cc} -0.909 & -0.023 \\ -0.359 & -1.685 \\ \hdashline -1.980 & -3.139 \\ \hdashline -0.970 & -0.032 \\ -0.032 & -0.949 \end{array}\right]$$

In comparison with Fig. 3, this smaller δ yields smaller unsafe regions as $\tilde{\Omega}_{u,1} \triangleq \mathcal{N}(\Omega_{u,1}, 0.409)$, $\tilde{\Omega}_{u,2} \triangleq \mathcal{N}(\Omega_{u,2}, 0.409)$ and $\tilde{\Omega}_{u,3} \triangleq \mathcal{N}(\Omega_{u,3}, 0.409)$. By the results in Fig. 4, we can conclude the safety of the uncertain switched system.

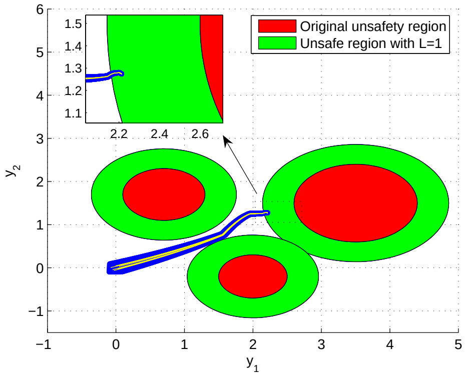


Fig. 3. The safety verification via SpaceEX for the certain system derived with ($L = 1$). The blue area is the reach set computed by SpaceEX, and the yellow lines are the random state trajectories. The safe or unsafe property of the original uncertain system cannot be concluded since the reach set of the certain system intersects with the new unsafe region.

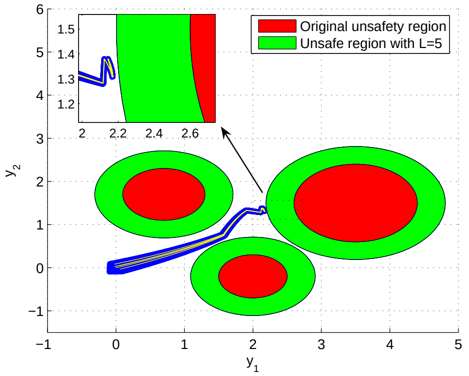


Fig. 4. The safety verification via SpaceEX for the certain system derived with ($L = 5$). The safety of the original uncertain system can be concluded since the reach set of certain system has no intersection with the new unsafe regions.

V. CONCLUSIONS

In this paper, the output reachable set estimation problem for switched linear systems has been investigated. With the aid of the common Lyapunov function and multiple Lyapunov function approaches, the output reachable set can be over-approximated by a set of bounding ellipsoids. Moreover, a sufficient condition for the existence of an approximate bisimulation of two switched linear systems is proposed, which can be viewed as an output reachable set estimation for the system combining the two bisimilar systems. Finally, by the result of approximate bisimulation, the safety verification problem for uncertain switched systems can be dealt with by verifying the safety of its bisimilar system with exact parameters. In this paper, A_i are required to be Hurwitz stable. By the techniques used in [33], the result in this paper can be readily extended to the case with some A_i are unstable. In addition, according to Table I, the computational cost significantly increases as the system order and number of modes grows, how to reduce the computational complexity and make it applicable for high dimensional systems with large amounts of subsystems will be our future study.

References

[1] R. DeCarlo, M. S. Branicky, S. Pettersson, and B. Lennartson, "Perspectives and results on the stability and stabilizability of hybrid systems," *Proceedings of the IEEE*, vol. 88, no. 7, pp. 1069–1082, 2000.

[2] H. Lin and P. J. Antsaklis, "Stability and stabilizability of switched linear systems: a survey of recent results," *IEEE Trans. Autom. Control*, vol. 54, no. 2, pp. 308–322, 2009.

[3] D. Liberzon, *Switching in Systems and Control*. Springer Science & Business Media, 2012.

[4] H. Yang, B. Jiang, and V. Cocquempot, *Stabilization of Switched Nonlinear Systems with Unstable Modes*. Switzerland: Springer, 2014.

[5] J. C. Geromel and P. Colaneri, "Stability and stabilization of continuous-time switched linear systems," *SIAM J. Control Optimiz.*, vol. 45, no. 5, pp. 1915–1930, 2006.

[6] Z. Sun, S. S. Ge, and T. H. Lee, "Controllability and reachability criteria for switched linear systems," *Automatica*, vol. 38, no. 5, pp. 775–786, 2002.

[7] L. Zhang and P. Shi, "Stability, ℓ_2-gain and asynchronous control of discrete-time switched systems with average dwell time," *IEEE Trans. Autom. Control*, vol. 54, no. 9, pp. 2192–2199, 2009.

[8] L. Zhang, S. Zhuang, and P. Shi, "Non-weighted quasi-time-dependent $\mathcal{H}_\infty$ filtering for switched linear systems with persistent dwell-time," *Automatica*, vol. 54, pp. 201–209, 2015.

[9] G. Zhai, B. Hu, K. Yasuda, and A. N. Michel, "Disturbance attenuation properties of time-controlled switched systems," *J. Franklin Inst.*, vol. 338, no. 7, pp. 765–779, 2001.

[10] E. Fridman, A. Pila, and U. Shaked, "Regional stabilization and $\mathcal{H}_\infty$ control of time-delay systems with saturating actuators," *Int. J. Robust Nonlin. Control*, vol. 13, no. 9, pp. 885–907, 2003.

[11] J. Lygeros, C. Tomlin, and S. Sastry, "Controllers for reachability specifications for hybrid systems," *Automatica*, vol. 35, no. 3, pp. 349–370, 1999.

[12] C. Durieu, E. Walter, and B. Polyak, "Multi-input multi-output ellipsoidal state bounding," *J. Optim. Theory Appl.*, vol. 111, no. 2, pp. 273–303, 2001.

[13] E. Fridman and U. Shaked, "On reachable sets for linear systems with delay and bounded peak inputs," *Automatica*, vol. 39, no. 11, pp. 2005–2010, 2003.

[14] J.-H. Kim, "Improved ellipsoidal bound of reachable sets for time-delayed linear systems with disturbances," *Automatica*, vol. 44, no. 11, pp. 2940–2943, 2008.

[15] Z. Feng and J. Lam, "An improved result on reachable set estimation and synthesis of time-delay systems," *Applied Mathematics and Computation*, vol. 249, pp. 89–97, 2014.

[16] P. T. Nam and P. N. Pathirana, "Further result on reachable set bounding for linear uncertain polytopic systems with interval time-varying delays," *Automatica*, vol. 47, no. 8, pp. 1838–1841, 2011.

[17] Z. Feng and J. Lam, "On reachable set estimation of singular systems," *Automatica*, vol. 52, pp. 146–153, 2015.

[18] Y. Chen, J. Lam, and B. Zhang, "Estimation and synthesis of reachable set for switched linear systems," *Automatica*, vol. 63, pp. 122–132, 2016.

[19] Y. Chen and J. Lam, "Estimation and synthesis of reachable set for discrete-time periodic systems," *Optim. Control Appl. Methods*, 2015.

[20] F. Amato, R. Ambrosino, C. Cosentino, and G. De Tommasi, "Input–output finite time stabilization of linear systems," *Automatica*, vol. 46, no. 9, pp. 1558–1562, 2010.

[21] F. Amato, M. Ariola, and C. Cosentino, "Finite-time stability of linear time-varying systems: analysis and controller design," *IEEE Trans. Autom. Control*, vol. 55, no. 4, pp. 1003–1008, 2010.

[22] A. Girard and G. J. Pappas, "Approximate bisimulation: A bridge between computer science and control theory," *European Journal of Control*, vol. 17, no. 5, pp. 568–578, 2011.

[23] G. Pola, A. Girard, and P. Tabuada, "Approximately bisimilar symbolic models for nonlinear control systems," *Automatica*, vol. 44, no. 10, pp. 2508–2516, 2008.

[24] A. Girard, "Controller synthesis for safety and reachability via approximate bisimulation," *Automatica*, vol. 48, no. 5, pp. 947–953, 2012.

[25] Y.-Y. Cao, J. Lam, and Y.-X. Sun, "Static output feedback stabilization: an ILMI approach," *Automatica*, vol. 34, no. 12, pp. 1641–1645, 1998.

[26] Z. Shu and J. Lam, "An augmented system approach to static output-feedback stabilization with $\mathcal{H}_\infty$ performance for continuous-time plants," *Int. J. Robust Nonlin. Control*, vol. 19, no. 7, pp. 768–785, 2009.

[27] A. S. Morse, "Supervisory control of families of linear set-point controllers part I. exact matching," *IEEE Trans. Autom. Control*, vol. 41, no. 10, pp. 1413–1431, 1996.

[28] L. Allerhand and U. Shaked, "Robust stability and stabilization of linear switched systems with dwell time," *IEEE Trans. Autom. Control*, vol. 56, no. 2, pp. 381–386, 2011.

[29] C. Briat and A. Seuret, "A looped-functional approach for robust stability analysis of linear impulsive systems," *Syst. Control Lett.*, vol. 61, no. 10, pp. 980–988, 2012.

[30] W. Xiang and J. Xiao, "Stabilization of switched continuous-time systems with all modes unstable via dwell time switching," *Automatica*, vol. 50, no. 3, pp. 940–945, 2014.

[31] W. Xiang, "Necessary and sufficient condition for stability of switched uncertain linear systems under dwell-time constraint," *IEEE Trans. Autom. Control*, vol. 61, no. 11, pp. 3619–3624, 2016.

[32] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, and O. Maler, "Spaceex: Scalable verification of hybrid systems," in *Computer Aided Verification*, pp. 379–395, Springer, 2011.

[33] W. Xiang, G. Zhai, and C. Briat, "Stability analysis for LTI control systems with controller failures and its application in failure tolerant control," *IEEE Trans. Autom. Control*, vol. 61, no. 3, pp. 811–816, 2016.

Appendix

Proof of Lemma 1: Define the following Lyapunov function as $V(t) = \sum_{i\in\mathcal{M}} \xi_i(t)V_i(x(t))$, where $\xi_i(t)$, $i \in \mathcal{M}$, is same as in Theorem 2. First, we consider any $t \in [t_k, t_{k+1}) \subset \mathcal{I}_i$, $\forall i \in \mathcal{M}$. (6) implies

$$\dot{V}(t) \le -\alpha V(t) + \alpha u^\top(t)R_u u(t),\ t \in [t_k, t_{k+1}) \tag{56}$$

Then, multiplying both sides of (56) with $e^{\alpha(t-t_k)}$ and then integrating it over $[t_k, t)$, we have $V(t) \le e^{-\alpha(t-t_k)}V(t_k^+) + \int_{t_k}^{t} e^{-\alpha(t-s)}u^\top(s)R_u u(s)\mathrm{d}s$. Due to $u(t) \in \mathcal{E}(R_u)$, $\forall t \in \mathbb{R}_{\ge 0}$, that is $u^\top(t)R_u u(t) \le 1$, $\forall t \in \mathbb{R}_{\ge 0}$, we have the following result

$$\begin{aligned} V(t) &\le e^{-\alpha(t-t_k)}V(t_k^+) + \int_{t_k}^{t} e^{-\alpha(t-s)}\mathrm{d}s \\ &= e^{-\alpha(t-t_k)}V(t_k^+) + 1 - e^{-\alpha(t-t_k)} \end{aligned} \tag{57}$$

and it can be rewritten to

$$V(t) - 1 \le e^{-\alpha(t-t_k)}(V(t_k^+) - 1),\ t \in [t_k, t_{k+1}) \tag{58}$$

Next, we consider $t_k \in \mathcal{S}$. From (7), we can obtain that $V(t_k^+) \le \beta V(t_k^-) + 1 - \beta,\ t_k \in \mathcal{S}$, which can be equivalently rewritten to

$$V(t_k^+) - 1 \le \beta(V(t_k^-) - 1),\ t_k \in \mathcal{S} \tag{59}$$

Combining (58) and (59), the following derivation can be obtained for $\forall t \in \mathbb{R}_{\ge 0}$

$$\begin{aligned} V(t) - 1 &\le e^{-\alpha(t-t_k)}(V(t_k^+) - 1) \le \beta e^{-\alpha(t-t_k)}(V(t_k^-) - 1) \\ &\le \cdots \le \beta^{\mathrm{Num}(t-t_0)}e^{-\alpha(t-t_0)}(V(t_0) - 1) \end{aligned}$$

where $\mathrm{Num}(t - t_0)$ denotes the number of switchings during $[t_0, t)$. Due to $\alpha > 0$ and $0 < \beta \le 1$, it means that

$$V(t) - 1 \le V(t_0) - 1,\ \forall t \in \mathbb{R}_{\ge 0} \tag{60}$$

Furthermore, (8) implies that $V(t_0) \le x_0^\top R_0 x_0 \le 1$, and (9) together with (60) yield that $y^\top(t)R_{i,y}y(t) \le V(t) \le 1$ holds when $\sigma(t) = i \in \mathcal{M}$, $t \in \mathbb{R}_{\ge 0}$. For all possible $i \in \mathcal{M}$, $y(t)$ thus satisfies $y(t) \in \bigcup_{i\in\mathcal{M}} \mathcal{E}(R_{i,y})$, $\forall t \in \mathbb{R}_{\ge 0}$ and therefore, $\mathcal{R}_y \subseteq \tilde{\mathcal{R}}_y$ by the definition of $\tilde{\mathcal{R}}_y$ given in (1).

8. List of Symbols, Abbreviations, and Acronyms

- CPS: cyber-physical system(s)
- DCPS: distributed cyber-physical system(s)
- Hynger: hybrid invariant generator
- HyperSTL: hyperproperties for signal temporal logic
- HyST: hybrid source transformation and translation software tool
- SLSF: Simulink/Stateflow
- StarL: stabilizing distributed robotics language
- STL: signal temporal logic
- RTA: runtime assurance
- UAS: unmanned autonomous system
- UAV: unmanned aerial vehicle